Universidade de Aveiro Departamento de Engenharia Mecânica
2019

NUNO GUILHERME ORNELAS ABRANTES

PLATAFORMA SEGWAY COM CAPACIDADES DE NAVEGAÇÃO AUTÓNOMA POR SEGUIMENTO DE LINHA

Universidade de Aveiro Departamento de Engenharia Mecânica
Ano 2019

NUNO GUILHERME ORNELAS ABRANTES

PLATAFORMA SEGWAY COM CAPACIDADES DE NAVEGAÇÃO AUTÓNOMA POR SEGUIMENTO DE LINHA

Relatório apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica do Doutor Miguel Armando Riem de Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e sob a coorientação científica da Doutora Ana Maria Pinto de Moura, Professora Auxiliar do Departamento de Economia, Gestão, Engenharia Industrial e Turismo.

“E falta sempre uma coisa, um copo, uma brisa, uma frase,e a vida dói quanto mais se goza e quanto mais se inventa.”

Álvaro de Campos

Aos meus avôs e avós.

o júri

presidente

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

vogais

Doutor Eurico Farinha Pedrosa
Bolseiro do Instituto de Engenharia Eletrónica e Telemática de Aveiro (arguente)

Prof. Doutor Miguel Armando Riem de Oliveira
Professor Auxiliar da Universidade de Aveiro (orientador)

Agradecimentos

Em primeiro lugar, quero agradecer aos meus pais que sempre me apoiaram durante o meu percurso escolar e sem eles nada disto era possível, à minha irmã por me chatear sempre nos momentos certos e aos meus verdadeiros amigos com os quais passei os últimos 5 anos.

Um agradecimento especial ao Professor Miguel Riem e à Professora Ana Moura pela ajuda, paciência com que orientaram o meu estágio e a importância dos seus conselhos.

Agradecer também a todos os meus professores que me acompanharam desde a pré-primária e me fizeram olhar o mundo pelas páginas da sabedoria.

E por último e não menos importante um enorme agradecimento a todos os meus colegas de trabalho da PERFO 4.0, operadores e responsáveis da Bout d'usine do Centro de Produção de Mangualde, por toda a ajuda e apoio que me deram ao longo do estágio e por todos os momentos que passamos dentro e fora do centro de produção.

Levo um pouco de vocês todos, para a vida.

Palavras-chave

Segway, Visão, Obstáculos, Iluminação, Reflexos, Controlador, Algoritmos, Ensaios, Câmara

Resumo

O presente trabalho enquadra-se nas áreas de condução autónoma, inteligência artificial e visão computacional. O Segway é um dos veículos adquirido pelo departamento de qualidade (*QCP*) do Centro de Produção de Mangualde (CPMG, que pertence ao *Grupo PSA*), para transportar operadores ao longo de um determinado percurso. Mas é necessário que o Segway, após efetuado o percurso com o operador, retorne ao seu lugar de origem de forma autónoma seguindo uma ou mais linhas.

Muito tem sido o trabalho desenvolvido nos últimos anos por parte do CPMG, desde o algoritmo criado para a navegação autónoma do Segway até à estrutura mecânica que torna possível a sua locomoção. No entanto, o Segway ainda não atingiu o seu melhor desempenho pois padece de inúmeras falhas.

Posto isto, neste relatório são determinados e abordados os principais problemas que afetavam o normal funcionamento do equipamento; são também propostas soluções para os problemas encontrados e, no fim, quantificadas e qualificadas estas mesmas propostas quanto ao seu impacto no funcionamento do Segway.

Desenvolveu-se um novo algoritmo para segmentação das linhas, pelas quais o Segway se deverá guiar, um algoritmo para controlo de direção e um algoritmo para deteção de obstáculos e consequente navegação autónoma, ao longo do trajeto a percorrer.

keywords

Segway, Vision, Obstacles, Lighthing, Reflexes, Controller, Algorithms, Essay, Camera

Abstract

The present work fits in the areas of autonomous conduction, artificial intelligence and computational vision. The Segway is one of the vehicles purchased by the quality department (QCP) of the Mangualde Production Center (CPMG, which belongs to the PSA Group) to transport the operators along a certain route. But it is necessary that the Segway be returned to its place of origin in an autonomous way by one or more steps.

The work was developed in the last years by the CPMG, since the algorithm created for an autonomous navigation in the Segway until the mechanical structure that makes possible its locomotion. However, the Segway has not yet the best performance.

That said, in this report are determined the main problems during the normal working of the equipment. The amounts of defefts were quantified and qualified for their impact on the operation of the Segway.

At same time was developed a new algorithm for line segmentation (through which must lead the way for the Segway), a direction control algorithm and an algorithm for the detection of obstacles and consequent autonomous navigation, along the path of the route.

ÍNDICE

I. ÍNDICE DE FIGURAS

II. ÍNDICE DE TABELAS

III. GLOSSÁRIO

Perfo 4.0 - Performance 4.0

CPMG - Centro de Produção de Mangualde

QCP - Qualidade do Centro de Produção

PCB - *Printed Circuit Board* (Placa de Circuito Impresso)

RMP - *Robot Mobility Platform* (Plataforma de Mobilidade Robótica)

HT - *Human Transport* (Transporte Humano)

FPS - Frame por segundo

Controlo PID - Controlo Proporcional, Integral e Derivativo

Controlo PD – Controlo Proporcional e Derivativo

Lidar - *Light Detection and Ranging*

BTU - *Bout D'ousine* (Final de linha na fábrica)

Sensor LDR - Sensor *Light Diode Resistor*

EPI's - Equipamentos de Proteção Individual

TMBF - Tempo médio entre falhas.

RGB - *Red, Green and Blue*

HSV - *Hue, Saturation and Value*

HSL- *Hue, Saturation and Luminosity*

YCbCr - espaço de cores para formato digital

LAB - *Luminosity, A (green and magenta), B (blue and yellow)*

PDCA - *Plan, do, check, act* (Planear, Desenvolver, Verificar e Ajustar)

Turno da Manhã - Turno que opera das 7 horas às 15 horas do dia.

Turno da Tarde - Turno que opera das 15 horas do dia às 23 horas da noite.

Turno da Noite - Turno que opera das 23 horas da noite às 7 horas do dia.

HD - *High Definition*

CEO - *Chief Executive Office*

1. INTRODUÇÃO

1.1. CONTEXTO

O presente estágio enquadra-se no ramo da indústria 4.0, condução autónoma e visão artificial. Desde o início do século que o Grupo PSA apostou nas novas tecnologias que emergem e antecipou-se à evolução dos mercados [(1)]. Por isso, no CPMG (Centro de Produção de Mangualde), é criada no ano de 2015 uma equipa de trabalho designada por Perfo 4.0, cujo objetivo foi "(...) tornar o grupo competitivo, desenvolvendo soluções de Kaizen Digital para apoiar a optimização dos processos de produção, atingindo a melhor performance e a máxima eficácia da fábrica (...)" [(2)].

Em 2015, nasce o projeto "*Segway*", que dá o contexto ao presente estágio. O projeto "*Segway*" veio ao encontro das necessidades do QCP (Qualidade do Centro de Produção). Os operadores da BTU (*Bout D'ousine*, zona pertencente ao QCP que trata da qualidade dos veículos no final da linha de montagem) têm de percorrer diáriamente km's, que lhes provocam fadiga e perdas de tempo na produção no final de um turno de trabalho. A plataforma "*Segway*", pode solucionar estes problemas, viabilizando o transporte dos operadores e encurtando o tempo utilizado entre deslocações, assim como a distância que estes caminham diariamente [1].

Apesar deste projeto já ter alguns anos, existiam ainda problemas, que se encontravam por resolver e que serão abordados ao longo deste relatório.

1.2. OBJETIVOS GERAIS

O principial objetivo deste estágio curricular passou por dotar a plataforma Segway, já existente na fábrica da PSA em Mangualde (CPMG), de capacidades de navegação autónoma de forma a que fosse possível que esta regressasse de forma automática ao seu local designado de parqueamento, após ter realizado o percurso com o operador.

Numa ótica do ciclo PDCA (Planear, Desenvolver, Verificar e Ajustar), durante este estágio, pretendeu-se dar continuidade ao projeto, que até então tinha vindo a ser desenvolvido.

A primeira fase do estágio começou pela elaboração de um plano de trabalho (*ANEXO 7*) onde se definem as metas e objetivos a realizar durante o período em que se realiza o mesmo. Destas metas e objetivos destacam-se as seguintes:

- Melhorar a performance atual do Segway ;
- Desenvolver um novo algoritmo de deteção de linha ou melhorar o atual;
- Implementar um sistema para deteção de obstáculos;
- Voltar a pôr o Segway em funcionamento na linha de produção aumentando o TMBF (Tempo médio entre falhas);

[1] Groupe PSA Homepage, www.groupe-psa.com, 2017
[2] PSA Groupe, Perfo 4.0: A Indústria do futuro ao serviço da Performance, 2019

A longo prazo, o segway deverá ser capaz de:

- Movimentar-se autonomamente através de análise do meio envolvente;
- Interagir com o ser humano, reconhecendo a sua presença;
- Reconhecer bem as suas tarefas de modo a ser competitivo com os humanos.

Como objetivo principal, está também a segurança. Esta deve estar constantemente presente, de modo a que não seja posto em risco, em nenhum momento, a saúde humana e em último caso, a integridade dos componentes do equipamento.

1.3. MOTIVAÇÃO

Desde que o Segway foi posto a circular na linha de produção têm vindo a ocorrer vários problemas, desde colisões com veículos, a quedas dos operadores durante a utilização do equipamento. Por estas razões, é importante analisar, repensar e agir com as melhores decisões para que acontecimentos como estes não se voltem a repetir.

Para além de trazer maior comodidade para os operadores e reduzir os tempos nas operações [1], este projeto, ao atingir as performances desejadas, poderá ser um grande passo dado pelo CPDM na industrialização 4.0. A futura expansão deste projeto para as restantes fábricas do grupo poderá ser uma solução a adotar.

A nível pessoal, este projeto permite que eu desenvolva as minhas capacidades de gestão e organização de tempo, de programação, automação em meio industrial e navegação autónoma de veículos.

1.4. ORGANIZAÇÃO DO DOCUMENTO

No capítulo número dois, é introduzido o *Grupo PSA* e a entidade acolhedora do estágio, o CPMG. É feita uma contextualização do projeto, do modelo do "*Segway*" utilizado e enumerados os problemas que o Segway apresenta.

No terceiro capítulo, são abordados alguns casos de estudo, que também se enquadram na temática da condução autonoma e alguns métodos utilizados para resolver os problemas definidos no capítulo anterior (capítulo dois).

No quarto capítulo, são caracterizados os problemas encontrados e as propostas/soluções são explicadas e postas em prática. Algumas destas soluções têm como base fundamental a pesquisa ciêntifica feita no capítulo número três. São ainda desenvolvidos três algoritmos, um de segmentação da linha, outro para deteção de obstáculos e outro para navegação e orientação autónoma do *Segway*. No quinto capítulo, são mostrados os resultados que se obtiveram e é analisado o impacto que as diversas alterações implementadas tiveram no desempenho final do Segway durante o seu percurso autónomo. Por fim, no último capítulo (sexto capítulo), é feita a análise crítica aos resultados obtidos no capítulo número cinco, indicando as falhas, objetivos que foram atingidos e os objetivos que ficaram por cumprir, mas que poderão servir de mote para trabalhos futuros.

2. O PONTO DE PARTIDA

Neste capítulo é introduzido o *Grupo PSA* e a entidade acolhedora do estágio, o *CPMG*. Numa segunda fase será feito o enquadramento do projeto *Segway*, nomeadamente os algoritmos, meios mecânicos e eletrónicos que possibilitam a locomoção autónoma do Segway. No final do capítulo será dado enfase às falhas que o Segway apresentava e que se pretende resolver com a realização deste estágio.

2.1. BREVE DESCRIÇÃO DO GRUPO PSA E DO CPMG

Com mais de 200 anos de história no mundo industrial e automóvel, tudo começou em 1810, quando os irmãos Jean-Peierre e Jean-Frédéric Peugeout embarcaram na indústria metalúrgica e automóvel. Surgiu então o primeiro veículo a gasolina de marca Peugeot em 1890 (*O tipo A*). No século seguinte, foi fundado o *Grupo PSA* (Peugeot Sociedade Anónima), em 1966. Só dez anos mais tarde é que se fez a fusão da *Citroën SA* com a *Peugeot SA,* dando origem ao grupo *PSA Peugeot Citroën*. Em 2016 o grupo *PSA Peugeot Citroën* passa a ser denominado *Grupo PSA.* Atualmente encontra-se espalhado por todos os recantos do mundo com excelentes prespetivas para o futuro [(3)].

A *Peugeot Citröen Automóveis Portugal, S.A* faz parte do grupo francês PSA, onde o atual CEO é o português Carlos Tavares. Este grupo caracteriza-se pela paixão[(3)] pelo produto automóvel.

Ao atingir uma dimensão internacional, com fábricas espalhadas por todo o mundo, o *Grupo* procurou incessantemente a flexibilidade, agilidade e eficiência para conseguir responder às constantes mudanças e características dos diferentes mercados onde está inserido. A adaptação e antecipação às novas tecnologias e novas aplicações que emergem nos mercados é uma preocupação e, por isso, ultimamente, o grupo tem apostado nos jovens, na investigação universitária e científica. O Grupo é o segundo maior construtor na Europa constituído por marcas das quais se referem a *PEUGEOT, CITROËN, DS AUTOMOBILES, OPEL e VAUXHALL*, tendo na liderança do mercado de veículos ligeiros uma quota de 20,2%[(3)].

Com cerca de 1000 colaboradores, o CPMG é uma das fábricas do *Grupo PSA* localizadas na Peninsula Ibérica. Este centro de produção é responsável pela produção dos modelos Peugeot Partner/Rifter, Citroën Berlingo/ Berlingo Van e, já este ano de 2019, irá ser iniciada a produção do novo Opel Combo/Combo Furgão. Por dia, nesta fábrica são fabricados 321 veículos e a sua produção está dividida em 3 turnos (turno da manhã, turno da tarde e turno da noite) [(4)].

A fábrica está dividida em várias unidades (*Figura 1*). O fluxo de produção passa por várias etapas, iniciando-se na Ferragem, passando depois pela Pintura, de seguida a Montagem e por fim a Qualidade. Estas quatro áreas são fornecidas pela Logística que faz a gestão de entrega do material numa ótica *just in time*. De referir que toda a produção funciona por encomenda, ou seja, todos os veículos que são produzidos já estão à partida vendidos a um cliente final [(4)].

[3] Groupe PSA Homepage, www.groupe-psa.com, 2017
[4] Groupe PSA Mangualde HomePage, https://site.groupe-psa.com/mangualde/pt-pt/, 2017

A missão principal desta entidade fabril é tripla [5]:

- Produzir com resultados tão bons ou equivalentes a outras fábricas do grupo.
- Satisfazer o cliente final com um veículo que responde inteiramente às definições “standard” ou específicas, respeitando as condições de QUALIDADE, PREÇO E PRAZO.
- Ser o modelo e a referência junto dos montadores do mundo inteiro, que estão relacionados com o Grupo PSA.

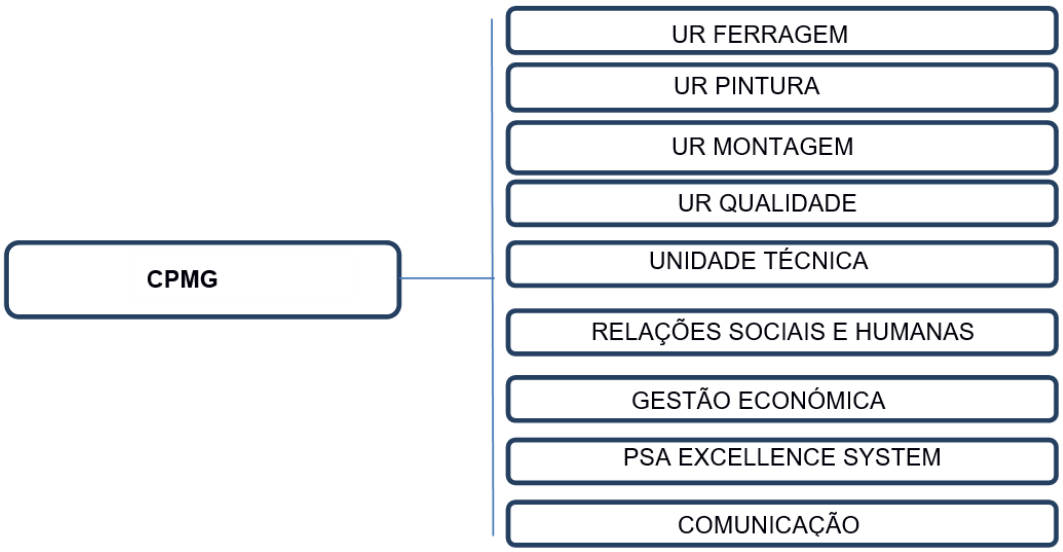


Figura 1 - Organigrama do CPMG [5]

2.2. CONTEXTUALIZAÇÃO DO PROJETO “SEGWAY”

O projeto do Segway é um projeto “made in CPMG”. Nesta secção pretende-se enquadrar o leitor com todo o trabalho que foi feito antes do estágio ser realizado. Aborda-se numa primeira instância o modelo da marca *Segway* que o CPMG adquiriu (*Segway i2 SE*) e os seus princípios básicos de utilização e funcionamento. É também feita a caracterização (pré-processamento, processamento, pós-processamento e interface com o utilizador) do algoritmo de deteção de linha, que, até então, estava a ser utilizado.

2.2.1. O SEGWAY I2 SE

A *Segway HT* apresentou em dezembro de 2001 uma scooter de duas rodas. Uma inovadora ideia para o mercado dos veículos de mobilidade humana. É um meio de transporte atrativo, com autonomia para percorrer alguns quilómetros [2].

Uma vez que só tem dois pontos de apoio com o solo, o Segway tem de estar num constante equilíbrio para se manter estável, e é este ponto que o diferencia de outros meios de transporte. O

[5] Manual de Introdução ao CPMG, 2019

Segway é um robô com balanceamento dinâmico, constituido por microcontroladores, baterias e motores DC. É um veículo desenvolvido para ser utilizado quer em ambientes *indoor* e *outdoor* [6]. Na Figura 2 e Figura *3*, podemos ver o princípio básico para manobrar este tipo de equipamento. O Segway funciona à base do equilíbrio e da inclinação que o operador estabelece com o equipamento. Quanto maior a inclinação maior a velocidade. As duas rodas são independentes uma da outra, o que permite que o Segway mude de direção quando estas giram a velocidades diferentes. A velocidade máxima que o Segway consegue atingir é de 20km/h e tem autonomia para mais de 8h seguidas de funcionamento. Estas e outras características podem ser encontradas na *Tabela 1*.

O Segway adquirido pelo *CPMG* é o modelo Segway i2 SE [6] (*Figura 4*). O sistema vendido pela marca *Segway* não é *open source*, o que impede o acesso ao sistema e aos algoritmos de controlo.

Este equipamento não está desenvolvido para se mover autonomamente, no entanto, verifica-se que depois de ligado pelo operador, este consegue mover-se a uma velocidade de 2km/h, com uma inclinação máxima de 10 ± 5º. Este facto possibilitou a oportunidade de dotar o Segway, adquirido pela fábrica, de capacidades de locomoção autónomas.

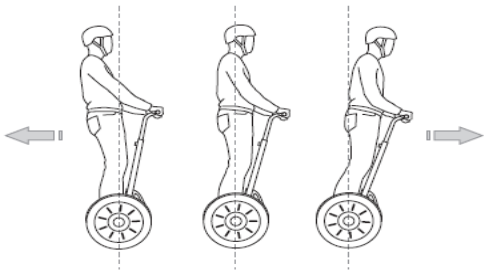


Figura 2 - Movimento de marcha atrás e movimento de avanço [6].

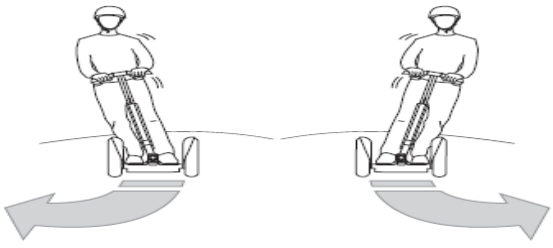


Figura 3 - Controlo de direção (virar à esquerda; virar à direita) [6].

[6] Segway Inc, Manual do utilizador “Segway Personal Transport”, 2014

Figura 4 - *Segway i2 SE* adquirido pelo CPMG [6].

Peso (Kg)	37
Altura (cm)	130
Comprimento (cm) x Largura(cm)	63x65
Diâmetro externo das rodas (cm)	48.3
Velocidade Máxima (km/h)	20
Alimentação	Bateria Li-ion, 73.6 V 5200 mAh

Tabela 1 - Características do Segway i2 SE [6].

2.2.2. CARACTERIZAÇÃO DO SISTEMA MECÂNICO E ELETRÓNICO

O Segway está equipado com uma câmara e um conjunto de outros componentes mecânicos. A câmara usada é uma PiCamera V2 (*Figura 6*). Para além de ser compacta, é possível controlar a sua resolução (da imagem capturada), exposição, velocidade de disparo e campo de profundiade (FoV) da câmara. Tudo isto e muitos outros aspetos podem ser definidos com o módulo de controlo disponível online para a PiCamera [(7)]. Na Tabela 2 podemos encontrar uma série de parâmetros característicos deste equipamento.

Sensor de imagem	Sony IMX 219 PQ CMOS
Tamanho (mm)	25 x 24 x 9
Resolução (Mpix)	8 (3280 x 2464)
Máxima taxa de transferência de imagem	1080p a 30fps
	720p a 60fps
Modo de coneção com o Raspberry	CSI connector
Livraria para controlo	Picamera Python library

Tabela 2 - Características da câmara PiCamera V2 [(7)]

O microprocessador utlizado é o *RaspberryPi 3 B+* [(8)] (*Figura 7*). É neste microcomputador que se realizam as tomadas de decisão para as ações levadas a cabo pelos atuadores mecânicos e onde são armazenados os dados e leituras necessárias. Na Tabela *3* podemos ver algumas das suas principais características.

Processador 1.4GHz 64-bit quad-core (ARMv8)
300Mbit/s ethernet
1GB de RAM
4 x USB 2.0 portas
Full size HDMI 1.3a port
Câmara interface (CSI)
MicroSD slot

Tabela 3 - Características do *RaspberryPi 3 B+* utilizado [(8)].

[7] D.Jones, “Picamera Release”, https://picamera.readthedocs.io/en/release-1.12/fov.html ,2013
[8] Raspberry Pi, Module Datasheet

A nível mecânico, o Segway foi apetrechado com um suporte onde estão apoiados os vários atuadores mecânicos [9] (*Figura 11*) que implementam as tomadas de decisão definidas pelo algoritmo. Ao todo são três os atuadores. Dois atuadores horizontais que permitem dirigir o Segway para a esquerda ou para a direita e um atuador vertical que controla a inclinação da haste do Segway e assim aumenta ou diminui a velocidade com que se desloca. As principais características dos atuadores estão enumeradas na *Tabela 4* e na *Figura 5*. A velocidade, precisão e comprimento de curso máximo e mínimo de cada atuador linear é controlada com recurso a um microcontrolador [10] (*Figura 8*). Para tornar os movimentos mais suaves, diminuindo os atritos entre a extremidade do atuador e o solo é acopolado na ponta do atuador um castor [11] (unidade mecânica constituída por uma esfera metálica, através da qual a ponta do atuador deslisa no pavimento) (*Figura 9*).

Potênciometro de 10 KΩ
Força máxima (sem carga) de 670 N
Velocidade máxima (sem carga) de 12mm/segundo
Alimentação de 12V

Tabela 4 - Características do Actuador Linear [9]

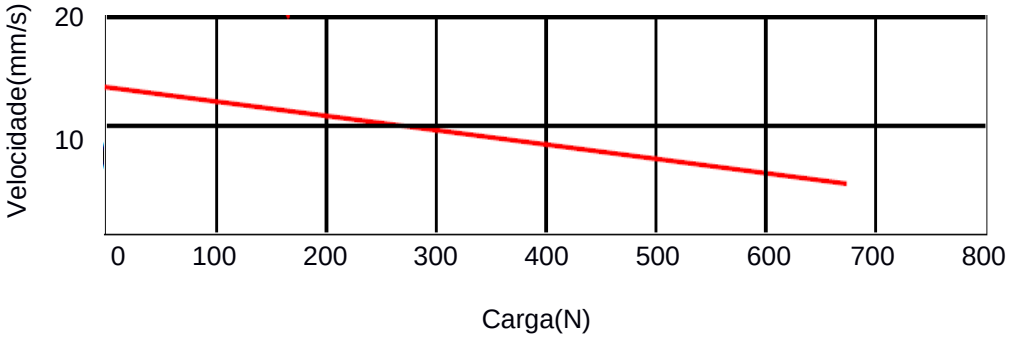


Figura 5 - Velocidade do atuador em função da carga que está sujeito [9]

[9] Atuador Linear FA-PO-150-12-XX, Firgelli Automation, “Product information”
[10] Devices, Actuonix Motion, “Actuonix Linear Actuator Control Board”
[11] Group RexRoth Boch, Ball Transfer Units Catalog

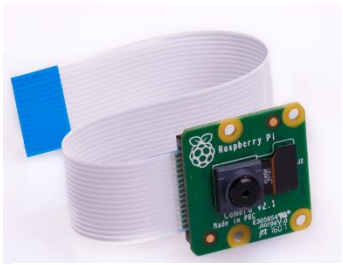


Figura 6 - Picamera V2 [7]

Figura 7 - RaspberryPi 3 B+ [8]

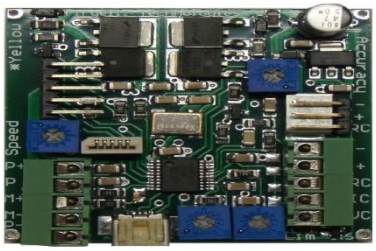
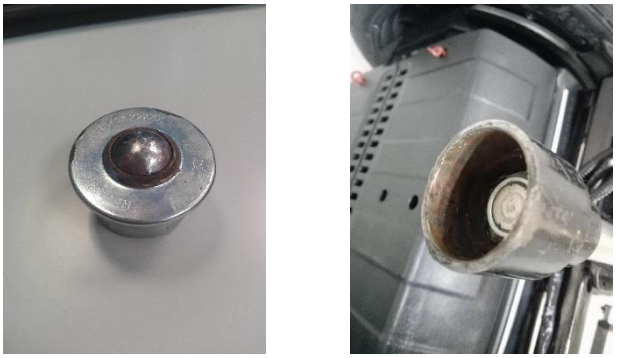


Figura 8 - Placa de controlo LAC [10]



a) b)

Figura 9 - (a) Castor [11]. (b) Base de suporte para o castor.

De modo a tornar todas as ligações organizadas e ter um o esquema elétrico simples, foi construída uma placa de circuito impresso (PCB), onde se interligam todos os dispositivos (deste as baterias que alimentam o circuito, aos leds que indicam o estado do processo ao operador e os atuadores que ajudam no controlo de velocidade e direção). Todos estes componentes são alimentados por uma bateria, das duas que constituiem o Segway. Dado que cada bateria dispende uma voltagem próxima de 74V e os componentes que a ela estão ligados não necessitam de mais do que 12V para serem alimentados, foi colocado um conversor de tensão, para que os componentes, que se alimentam da bateria não fossem sobrecarregados.

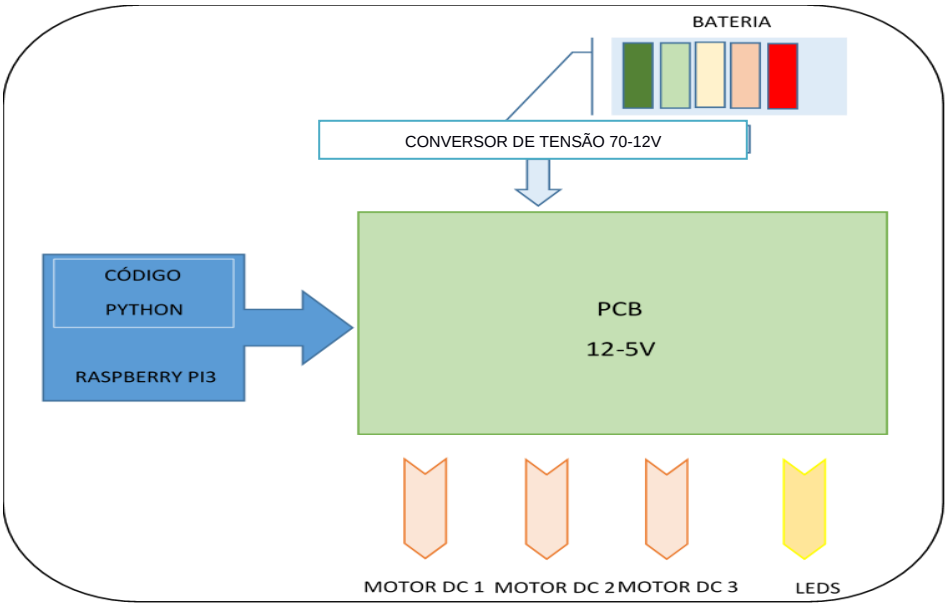


Figura 10 - Arquitetura eletro-mecânica do Segway.

Figura 11 - Atuador Linear [9]. Referência: **FA-PO-150-12-XX**

2.2.3. CARACTERIZAÇÃO DO ALGORTIMO DE VISÃO

Para guiar o Segway, está colocada no lado esquerdo do percurso uma fita marcadora de pavimento de cor amarela e uma fita de cor verde (*Figura 12*). Com uma *PiCâmara V2,* as frames são capturadas com uma resolução de 720p (HD), a uma velocidade média de 6 FPS e posteriormente pré-processadas e processadas pelo *RaspBerryPi 3 B+.* Um interruptor localizado na haste do Segway, que permite comutar entre o modo manual (onde o Segway precisa do operador para se movimentar) e o modo autónomo (onde o Segway se movimenta autonomamente) (*Figura 13*). Esta mudança de estado, quando o operador comuta o interruptor, altera o estado de uma variável booleana denominada (*input_state*).

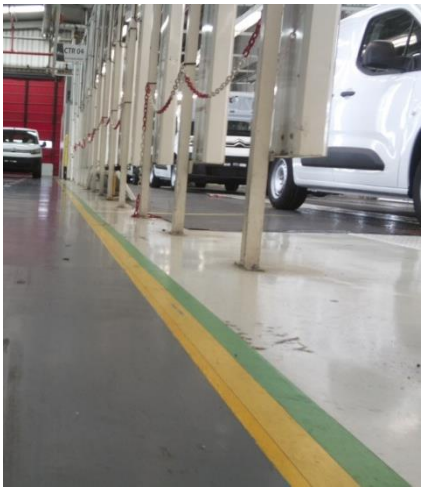


Figura 12 - Linhas marcadoras do solo.

a) b)

Figura 13 - Modos de operação; (a) Modo Manual; (b) Modo Autónomo;

Para a análise das imagens é utilizada a biblioteca de programação, disponível online, *OPENCV* [12]. O algoritmo foi elaborado com recurso ao sistema operativo Linux-Ubuntu que vem instalado no *RaspberryPi 3 B+*. A linguagem de programação utilizada para elaborar o algoritmo é a linguagem *Python*.

Sempre que o modo autónomo está acionado, é feito o pré-processamento. Durante o pré-processamento o *RaspeBerryPi* começa a ler as imagens, que anteriormente foram captadas pela câmara, num formato RGB. A imagem RGB é convertida para o formato LAB. É aplicado um filtro equalizador na camada L (luminosidade) para aumentar os contrastes na imagem entre a linha e o pavimento. Posteriormente a imagem passa de novo para o formato RGB, onde de seguida é aplicado um filtro Gaussiano (com o intuito de atenuar o ruído). É também definida a região de interesse, cortando-se as zonas da imagem que não interessam, pois, a probabilidade de encontrar a linha amarela nestas zonas é pouca ou nenhuma (*Figura 14*-(a)). Terminado o pré-processamento, inicia-se o processamento da imagem com a segmentação das linhas num formato HSV. Os valores máximos e mínimos, para cada camada (H, S e V), utilizados para segmentar simultaneamente as linhas amarela e verde estão na *Tabela 5*. Deste modo é possível isolar as linhas de todo o resto da imagem (*Figura 14*-(b)).

Hmin	**Smin**	**Vmin**
31	60	60
Hmax	**Smax**	**Vmax**
105	255	255

Tabela 5 - Parâmetros HSV para segmentar as linhas amarelas e verde.

Estando feita a segmentação das linhas utilizando a cor, é aplicado um filtro detetor de arestas (*Canny*). Este filtro deteta os contornos da linha amarela e reprodu-los numa imagem binarizada (Figura *14*-(c)). Depois de encontrados os contornos, inicia-se o pós-processamento onde são feitas as medições e retiradas as variáveis para usar no controlo de direção. Pegando na imagem binarizada, é calculada a distância (*Equação 1*) em píxeis desde o centro da imagem REF (*Figura 14*-(d)) com coordenadas (*REF X, REF Y*) até ao primeiro pixel branco que se encontra na mesma linha da imagem (y=*REFX*) com as coordenadas (*xPixel, yPixel*), onde *REFX=xPixel* e na Figura *14*-(c) é ilustrado por *X*. É este valor de distância, calculado pela *equação 1*, que serve de entrada para o algoritmo que controla a direção do segway e permite definir o curso dos atuadores lineares horizontais. Se não for encontrado qualquer píxel com valores de H,S e V dentro do patamar estabelecido na Tabela *5* ou caso sejam encontrados pixeis, mas por algum motivo estes não tenham como coordenada y, y=REFX, então são dadas ordens para que o segway se imobilize.

[12] OpenCV team, https://opencv.org/, 2019

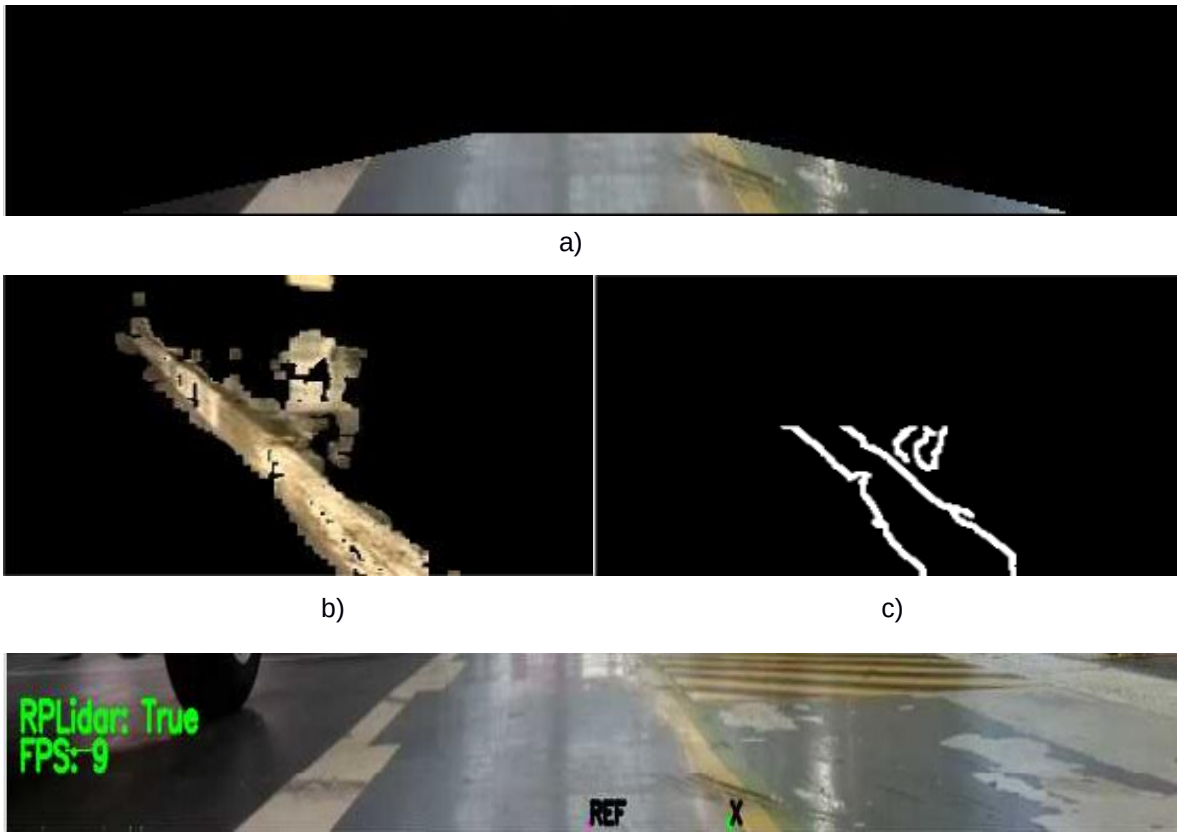
$$distância = \sqrt{dx^2 + dy^2} \quad (1)$$

Onde

$$dx = REFX - xPixel \quad (2)$$

$$dy = REFY - yPixel \quad (3)$$

No fluxograma da Figura *15* é resumido o algoritmo que controla o Segway.



a)

b) c)

d)

Figura 14 - (a) Pré-processamento. (b) e (c) Processamento. (d) Pós-processamento.

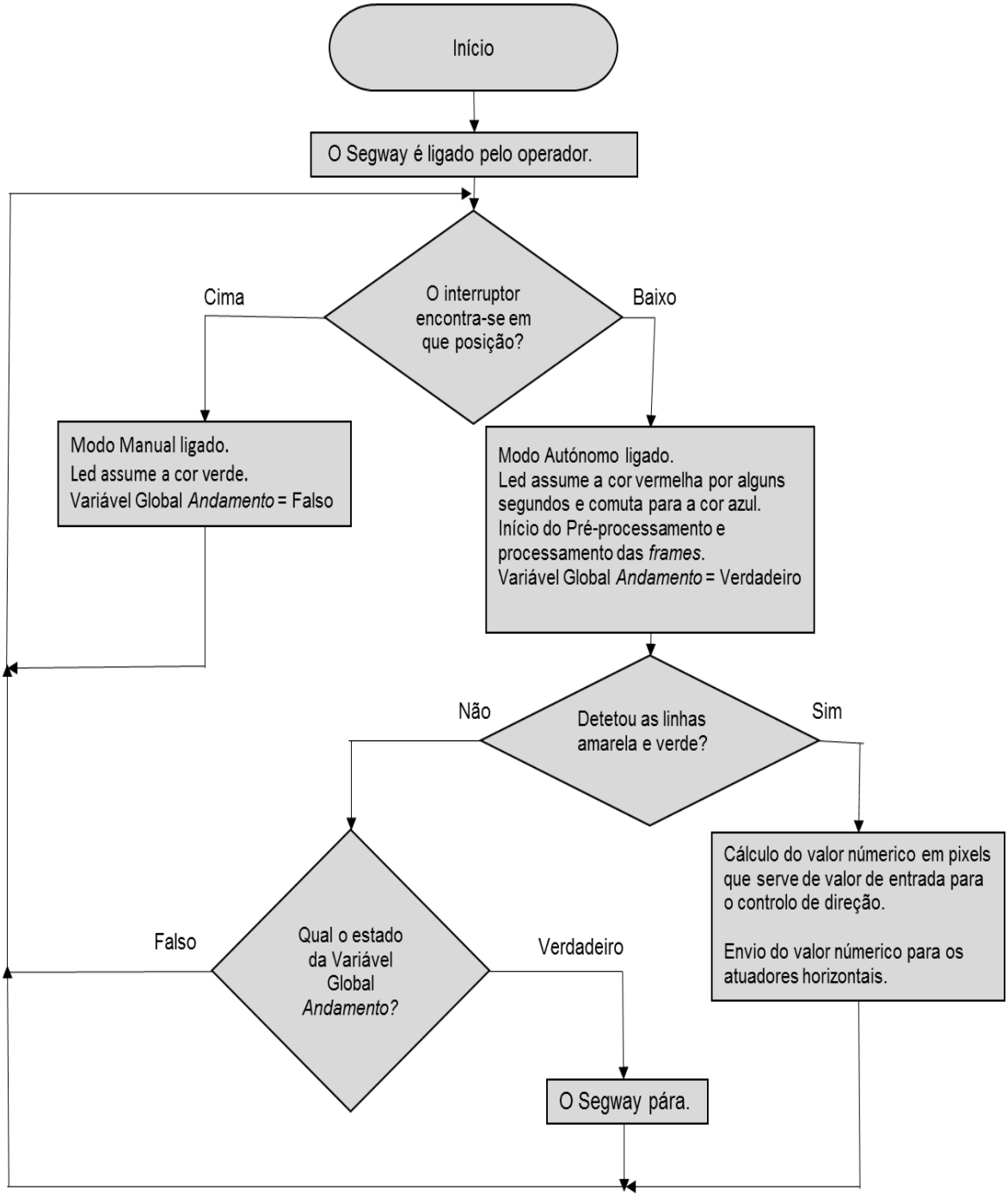


Figura 15 - Fluxograma do processo de deteção de linha.

2.2.4. CARACTERIZAÇÃO DA INTERFACE COM O UTILIZADOR

O desenvolvimento de uma interface permite que o utilizador analise e controle em tempo real uma série de variáveis. Uns dos parâmetros que é possível definir é o valor da cor a segmentar num formato HSV (*Figura 16*). Pode-se também definir o ponto de referência (REF X e REF Y) e assim ajustar em que linha queremos calcular o valor da distância (*Equação 1*, sub-secção 2.2.3) e também é possível definir a região de interesse na imagem (W. Y1, W. Y2, H.X1 e H.X2).

Para além dos parâmetros que são possíveis de controlar, também é possível analisar o estado e posição de certas variáveis em tempo real. No canto superior esquerdo da janela intitulada "*Capturada"* (*Figura 17*) é impresso o valor da taxa de processamento das frames em FPS. É também mostrado o estado do detetor de obstáculos, que ainda não está implementado no sistema. Na mesma janela (*Figura 17*) é feita a ilustração do ponto de referência a vermelho e a localização do pixel determinado pelo algoritmo para o cálculo final do valor da distância (*Equação 1*). São mostradas mais duas janelas, uma intitulada "*corte*" (*Figura 17*) que mostra a zona de interesse definida pelo programador e outra intitulada de "*Binarizada*" (*Figura 17*) onde se mostra os contornos das cores anteriormente segmentadas.

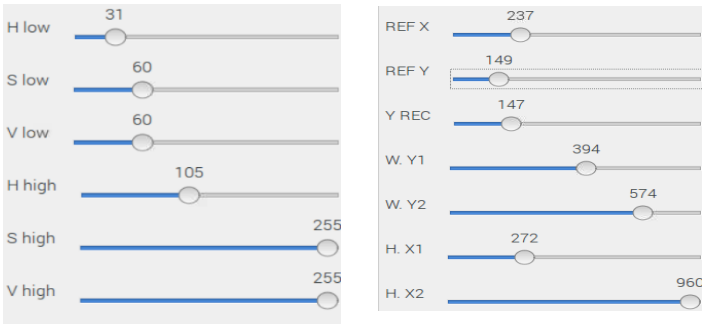


Figura 16 - Parâmetros controlados através da interface com o utilizador.

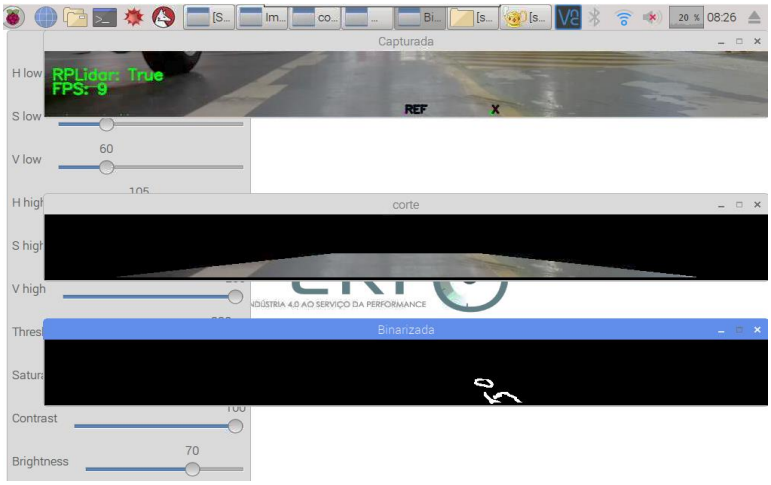


Figura 17 - Layout do Ambiente de Trabalho.

2.3. DEFINIÇÃO DO PROBLEMA GERAL

Como já foi dito anteriormente, o Segway tende a ser utilizado para reduzir os tempos de deslocamentos dos operadores do BTU, transportando-os de uma forma ergonómica, rápida e em segurança em tempo útil. No entanto, o percurso onde este é usado é unidirecional, no que diz respeito ao transporte de operadores.

É necessário que o Segway volte ao local de partida de forma autónoma e correta, após a saída do operador. O Segway transporta o operador do local C para o local A (*Figura 18*), mas não há operador que o possa trazer de volta para o local C, lugar onde se inicia o transporte do operador, como se ilustra na Figura 18. Por esta razão surgiu a ideia de retornar o Segway autonomamente para o ponto de partida. Aplicaram-se então os mecanismos e o algoritmo apresentados nas *sub-seções 2.2.1, 2.2.2 e 2.2.3*. No entanto, após alguns ensaios feitos ao longo dos vários turnos (turno da manhã, turno da tarde e turno da noite), o Segway não conseguia deslocar-se autonomamente.

O Segway já esteve a funcionar a tempo inteiro durante os vários turnos, mas devido às contantes falhas e problemas que causou, acabou por ser retirado do seu normal funcionamento, estando desde então parado nas instalações da fábrica. Os principais motivos para que este equipamento tenha sido retirado do seu pleno funcionamento, foram os seguintes:

- Trajetórias deficientes e irregulares;
- Frequentes colisões com humanos, veículos e outros objectos;
- Queda dos operadores durante a sua utilização.
- Baixo TMBF (Tempo médio entre falhas).

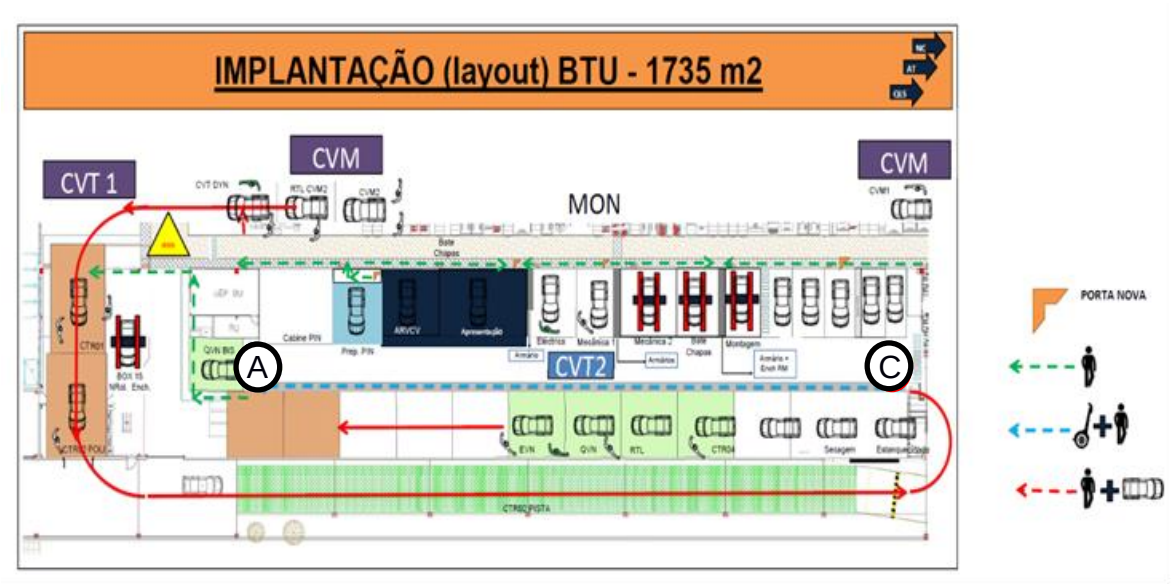


Figura 18 - Plano de Circulação do Segway na BTU; disponível também em anexo [*ANEXO 2*].

3. REVISÃO DA LITERATURA

No presente capítulo é abordada parte da pesquisa feita em teses, projetos, artigos e páginas web que possíbilitaram a aquisição de competências e de um leque de conhecimentos para a tomada de decisões face aos problemas que foram introduzidos no capítulo dois.

Numa primeira fase (*secção 3.1*) são apresentados alguns exemplos de aplicações que também envolvem a condução autónoma de protótipos ou de produtos já comercializados no mercado. Numa segunda fase (*secção 3.2*), são abordadas formas de corrigir os principais problemas que poderão afetar o Segway durante o seu percurso autónomo, sendo estes problemas a deteção de linhas, deteção de obstáculos e controlo de direção, em ambientes onde os fatores externos (luz, condições do piso, reflexos e sombras) são difíceis de controlar.

3.1. APLICAÇÕES NA ÁREA DA CONDUÇÃO AUTÓNOMA

3.1.1. O SEGWAY LOOMO

Este modelo desenvolvido pela *Segway* é dotado de navegação autónoma. Para além de servir como meio para transportar pessoas é capaz de interagir com os humanos e até mesmo seguir objetos.

Possui um processador *Intel Atom Z8750* com uma velocidade de 2.56GHz [13]. Para fazer a aquisição de imagem é dotado de uma câmara *Intel RealSense* com uma resolução de 1080p (*HD*), capaz de atingir uma frequência de frames de 30Hz, durante o modo vídeo. Estas caracteristicas permitem que o Loomo capture imagens com elevada qualidade criando todo um leque de interações com o utilizador [14]. O software utilizado é OpenSource dando liberdade ao utilizador para desenvolver uma série de aplicações com o equipamento. Para detetar obstáculos e evitar colisões o *Loomo* está equipado com um sensor que mede distâncias por ultrasons.

3.1.2. O SEGWAY RMP

Em maio de 2003 a Segway lança a plataforma de mobilidade robótica (*RMP*). Um grupo de investigadores da universidade de Carnegie, na Pennsylvania, juntou-se com o objetivo de tornar este tipo de plataforma capaz de jogar futebol autonomamente. Foi necessário repensar o modo de fazer a comunicação entre os equipamentos, como criar o modo do robô pensar e agir, entre outros desafios que estão documentados em [2].

É importante ter algoritmos de visão que processem rapidamente a informação para que, por exemplo, o robô tenha a perceção, em tempo real, das mudanças de iluminação no meio que o rodeia. No entanto, neste aspecto da iluminação, a maior parte das bibliotecas desenvolvidas não estão preparadas para mudanças de iluminação, que variam frequentemente [2]. Por isso, para detetar os objetos (neste caso a bola de futebol) em ambientes onde a iluminação pode ser um

[13] Segway inc, https://www.segwayrobotics.com/specs , 2019
[14] Segway inc, http://www.segway.ch/en/aktuelles/meldungen/Loomo.php , 2019

problema, dificultando a deteção dos mesmos, recorrem a um dos modelos de *segmentação por "region growing"* [3].

3.1.3. O LINE TRACER

A Leggo lançou um modelo que se assemelha a um Segway e que pode ser programado a partir de uma placa *NXT Brick* (microprocessador, que pode receber informação de sensores de toque, sensores de som, sensores de luz e sensores ultra sónicos) [4].

Adaptar este Leggo e com ele desenvolver um protótipo semelhante a um Segway, capaz de se mover autonomamente por seguimento de uma linha, foi um dos projetos que um grupo de investigadores da Universidade de Brawijaya (Indonésia) levou a cabo. O robô tomou o nome de Line Tracer. Este usa um sensor de cor para detetar a cor de uma linha e assim o robô sabe a direção que tem de seguir.

Em [5] desenvolveu-se um controlador de equilíbrio PID e o robô Line Tracer consegue assim adquirir equilíbrio sem necessitar do auxílio humano e seguir a linha sem cair no chão.

3.1.4. O FÓRMULA UM TD2

No ano de 2013, na Universidade do Minho, foram feitos trabalhos de modo a dotar o veículo *Fórmula UM TD2* com capacidades de locomoção autónomas [6]. Num destes trabalhos é desenvolvido um conjunto de algoritmos, que dotam o veículo com capacidade de realizar tarefas de forma autónoma com reconhecimento de sinalização, estacionamento e planeamento de trajetória [6].

Em relação à arquitectura de Software utilizado, os algoritmos têm como alicerce o sistema operativo *Linux Ubuntu* e a linguagem de programação é a linguagem *C++.*

Os algoritmos são construídos utilizando o ambiente de desenvolvimento de software *QT Creator*, com a ajuda de bibliotecas como o *OpenCV* (que ajudou na parte da visão computacional) e o *OMPL* (de onde é retirado o algoritmo para o planeamento de trajetória).

Para controlar os processos que comandam o veículo, usou-se um computador *HP Pavilion Dv6*, com processador *Intel Core 2 Duo* de 2.26GHz, que o autor considera com uma velocidade de processamento aceitável para a aplicação em prática [6]. Este computador comunica por porta série com os restantes equipamentos, nomeadamente um microcontrolador *(Arduino ATMega).* O veículo é equipado com uma câmara 3D, *câmara Kinect* [7], que é constituida por um sensor de profundidade, um microfone, um motor de inclinação, uma câmara RGB (com 640 x 480 de resolução a 30fps) e uma câmara IV (com sensor CMOS). A *câmara Kinect* permite uma leitura realista do ambiente onde se encontra, sendo útil para guiar autonomamente o veículo [6].

3.2. DESAFIOS NA CONDUÇÃO AUTÓNOMA

Após serem abordados na *secção 2.3* algumas situações que impossibilitam que o Segway complete o seu percurso, e, depois de se terem apresentados alguns casos práticos na *secção 3.1*, é necessário agora perceber que tipo de soluções e métodos são propostos pelos especialistas na área. Como o projeto *Segway* está diretamente relacionado com a área de condução autónoma, os métodos e soluções encontrados poderão ser uma hipótese para resolver alguns dos problemas que o Segway ainda enfrenta.

3.2.1. MÉTODOS PARA DETEÇÃO DE UMA OU MAIS LINHAS

Durante o processo de condução autónoma, a identificação das linhas, que delimitam o espaço a percorrer pelo veículo, pode ser vantajoso para que este se guie ao longo do trajeto a percorrer. O uso de bibliotecas como o *OpenCV* para analisar as imagens e para conseguir segmentar as linhas por onde o veículo deve seguir é bastante comum em algumas aplicações [(15), (16)].

Nos últimos anos, muitos têm sido os algoritmos de deteção de linhas, desenvolvidos para casos práticos de condução autónoma [8]. Mas será que estes algoritmos conseguem ultrapassar problemas como sombras, excesso de reflexos, pisos em mau estado, veículos que obstruam a linha a segmentar, entre outros fatores inesperados?

Antes de se iniciar o tratamento de uma qualquer imagem é importante definir um conjunto de pressupostos. Os quais se destacam:

- Nem toda a imagem contém informação útil;
- A maioria das linhas que marcam os pavimentos das vias rodoviárias ou são amarelas ou brancas;

Definir a região de interesse é uma das primeiras etapas a realizar, depois de capturada a imagem e após a leitura da frame/imagem pelo algoritmo a desenvolver. Isto é importante porque vão existir sempre zonas onde a probabilidade de encontrar uma linha ou outro objeto, que se pretende segmentar, é reduzida ou nula [(17)]. A região de interesse pode ser definida determinando o conjunto de pontos que fazem parte do plano de interesse (pode ser a estrada onde o veículo se desloca, por exemplo), com recurso a técnicas de estereoscopia [9], [10] ou câmaras 3D. Em outros casos [(18)], pode-se definir a zona útil cortando a imagem a duas dimensões.

Nas estradas, as linhas que delimitam as faxas de rodagem ou são brancas ou são amarelas [(18)]. Alguns autores têm preferência para analisar as imagens em tons cinza, no entanto, quando a cor do objeto é importante, deve-se adquirir a imagem com uma câmara *RGB* e analisar a imagem

[15] Shibuya Naoki, https://github.com/naokishibuya/car-finding-lane-lines, 2017
[16] Kemfic, https://www.hackster.io/kemfic/curved-lane-detection-34f771 ,2018
[17] Matt Hardwick,https://medium.com/@mrhwick/simple-lane-detection-with-opencv-bfeb6ae54ec0, 2017
[18] Percy Jaiswal,https://towardsdatascience.com/finding-driving-lane-line-live-with-opencv-f17c266f15db, 2018

num formato *RGB* [11]. Para segmentar os objetos através da cor, pode-se recorrer a diversos formatos como o *HSL* ou *HSV* [15]. As imagens podem ser analisadas também em *YCbCr*, *HSI* ou *LAB* [9], mas, no final, o importante é conseguir segmentar a linha ou objeto da melhor maneira possível.

Cada cor tem um valor de refletância que a caracteriza e que define a quantidade de luz que essa mesma cor consegue absorver. Existem algumas tabelas [19] disponíveis online onde se pode observar as diferentes refletâncias para diferentes tipos de cor e assim escolher qual a melhor cor a usar no tipo de ambiente em que a aplicação se enquadra.

Por vezes a linha não apresenta o melhor contraste com o meio onde se insere. Nestes casos, o aumento do contraste entre a linha e o meio que a envolve é um fator importante, para que a segmentação seja bem feita [12]. O uso de filtros equalizadores (como o *CLAHE* em formatos *HSV* ou *RGB*) podem ser uma boa solução para aumentar o contraste na imagem [13].

A aplicação de filtros que diminuem o ruído na imagem (filtros gaussianos e filtros de média) é uma solução recomendada por alguns autores para melhorar a imagem antes de ser feita a segementação da linha [9]. Tendo um bom contraste entre a linha e o meio envolvente, facilmente se faz a segmentação.

Depois de definida a região de interesse e segmentada a região que queremos encontrar, é o momento de se extrai alguns parâmetros (comprimento, largura, entre outros) da linha a detetar. A biblioteca *OpenCV* disponibiliza funções, que permitem detetar as linhas (*Canny*, o mais utilizado na deteção de arestas [18], e "*HoughLinesP",* para encontrar os pontos extremos das linhas na imagem [14]).

Na prática, nem sempre o melhoramento de imagem e posterior segmentação conduzem às melhores medições; os resultados finais podem trazer uma série de erros acomolados, que podem ser críticos, dependendo do nível da aplicação.

Quando falamos num algoritmo que corre dezenas de frames em alguns segundos, o efeito de alguns distúrbios (reflexos, sombras, danos no piso, entre outros) nas medições finais, podem ser eliminados aplicando um filtro de ruído. Filtro este que passa por limitar alguns parâmetros, como por exemplo, a inclinação máxima e mínima das linhas a detetar ou a média dos valores de um ou outro parâmetro obtido nas últimas frames em que a linha foi detetada. Isto permite que o novo valor a medir, caso a frame apresente alguma anomalia ou distúrbio, não seja tão afetado como seria se não fosse aplicado qualquer filtro [16].

O hábito mais comum e aconselhado, passa por testar o algoritmo com uma sequência de imagens e só, numa última etapa, testar o algoritmo numa sequência de frames (modo vídeo) [17]. Por estas razões, o modo como contruímos e desenvolvemos o algoritmo para a deteção de linha é importante para termos as melhores medições e consequente desempenho final da aplicação desenvolvida.

[19] Herman Miller Laminates, LRV (Light Reflectance Value)

3.2.2. CONTROLO DE AMBIENTES COM FORTES VARIAÇÕES DE LUZ

O modo como o espaço que envolve o objeto em conjunto com a câmara e o respetivo objeto são iluminados é um dos grandes obstáculos na visão autónoma das máquinas, uma vez que representa em média 80% dos problemas de visão. Muitos dos algoritmos que se constroem atualmente acentam em pressupostos de iluminação constante [2]. No entanto, quando temos uma aplicação que está desenvolvida para atuar num ambiente rodeado de humanos e é necessário interagir com os mesmos, onde as condições de luz no momento são imprevisíveis com sombras e reflexos, o problema aumenta de complexidade. Contudo, apesar de complexo, é possível ultrapassá-lo e muitos são os algoritmos que têm sido desenvolvidos [15], [16].

Numa primeira instância, as imagens capturadas pela câmara de um robô devem ser o mais simples possíveis, conferindo um bom contraste em torno do objeto a detetar [12]. O aumento de contraste pode conseguir-se encontrando a melhor posição geométrica entre a câmara, que captura a imagem, o objeto a ser detetado e a luz artificial que ilumina a ação.

Os problemas referentes à iluminação podem dividir-se em dois tipos: reflexos e sombras [17]. Das soluções propostas para ultrapassar estes obstáculos destacam-se os filtros (lentes) para aplicação nas câmaras, reduzindo assim a interferência da luz solar (filtros UV) e a interferência dos reflexos que insidem no sensor da câmara (filtros polarizadores). Algumas lentes já vêm com estes dois filtros implementados (por exemplo, as lentes dos óculos de sol). Nestes casos, a aplicação de técnicas, como o "*strobing*" com iluminação LED também pode ser ponderada [12]. Todas estas técnicas têm como objetivo final melhorar os contrastes e conseguir obter as melhores medições na aplicação desenvolvida, face às inúmeras mudanças de iluminação.

3.2.3. CONTROLOS AUTOMÁTICOS

Identificar a linha na imagem não é suficiente para que o robô siga a trajetória desejada. É necessário a existência de um controlo que leve o robô a tomar uma determinada ação. Esta ação é levada a cabo, em último nível, por atuadores mecânicos (motores elétricos, cilindros pneumáticos, entre outros).

O controlo automático tem como finalidade a manutenção de uma certa variável ou condição próxima de um valor referência [18].

Alguns sistemas têm dificuldade em encontrar a estabilidade e para cada aplicação em concreto é necessário um controlo diferente. Existem inúmeras aplicações de controladores no mundo industrial e muitas são as dicas teóricas [18], a fim de construir o melhor controlador para algumas aplicações, desde controladores "On-Off," controladores "PD" [19] e controladores "PID" [20], para tornar os sistemas estáveis em volta de um ponto de equilíbrio/referência.

Dentro dos controladores PID, existem três componentes. A componente proporcional que é diretamente proporcional ao erro (diferença entre o valor de referência e o valor real enviado para os controladores). A componente integral, que permite diminuir o "off-set" (diferença entre o valor de referência e o valor onde o controlador estabiliza) que eventualmente possa ocorrer entre o valor de

referência e o valor de equilíbrio. E a componente derivativa, que é usada para controlar a velocidade de reação consoante as mudanças que o sistema possa sofrer [21]. As componentes, proporcional (P), integral (I) e derivativa (D), apresentam todas elas vantagens e desvantagens [18]. Cabe ao utilizador perceber qual a melhor combinação para a sua aplicação, de modo a que se atinja um equilíbrio dentro de um regime estável.

3.2.4. DETEÇÃO DE OBSTÁCULOS

É importante que um veículo seja capaz de reconhecer que está perante um obstáculo para evitar possíveis colisões que possam pôr em risco a integridade do mesmo ou dos que o rodeiam (objetos e operadores). Após algumas pesquisas podem-se encontrar vários tipos de sensores, que são usados em diversas aplicações para medir distâncias e consequentemente evitar ou detetar objetos. A utilidade desdes sensores de distância varia consoante o alcance (distância) e o intervalo ângular a que pretendemos detetar o objeto. A eficácia de alguns sensores depende do tipo de superfície, cor e material de que é feito o objeto a detetar (como acontece com alguns sensores óticos). Muitos necessitam que haja contacto entre o objeto e o sensor para que ocorra a deteção (exemplo dos sensores Eletro-Mecâncios e Pneumáticos), enquanto que outros sensores especificam uma distância mínima a partir da qual conseguem detetar os objetos.

Existem sistemas que detetam obstáculos em tempo real, usando técnicas de visão estereoscópica [22]. Para o mesmo efeito, outros autores [23] usam sensores óticos (LIDAR) tendo estes a vantagem de medir distâncias num intervalo de 360º. Noutras aplicações é recorrente o uso de uma câmara 3D, para cálculo de distância e assim poder evitar obstáculos [7]. Os sensores ultra-sons também são muito utilizados no cálculo de distâncias em aplicações robóticas, tendo estes a vantagem de serem menos sensíveis ao tipo de superfície e cor dos objetos [24] quando comparados com alguns sensores óticos.

4. PROPOSTA E IMPLEMENTAÇÃO DA SOLUÇÃO

No presente capítulo são caracterizadas as variáveis que influenciam o desempenho do Segway e propostas soluções, com base nas aplicações e métodos abordados no capítulo três, para corrigir os problemas encontrados. Os resultados destas propostas serão demonstrados no capítulo cinco.

Neste capítulo (capítulo quatro) caracterizam-se os consumos de energia; apresentam-se as tabelas de manutenção e de regras a ter durante as operações com o Segway, para prevenir situações problemáticas e, consequentemente, aumentar o tempo médio entre falhas.

4.1. CARACTERIZAÇÃO DA ILUMINAÇÃO

No final da linha de montagem inicia-se o controlo de qualidade. Nesta zona final da fábrica (denominada Bout d'usine), pretende-se que o Segway se mova autonomamente desde o local A para o local C (*Figura 19*). Ao longo do seu percurso autónomo, o Segway tem de passar por diferentes zonas com diferentes tipos e quantidades de iluminação. Por isso, dividiu-se o percurso em três setores: o setor A (amarelo na *Figura 19*), o setor B (vermelho na *Figura 19*) e o setor C (azul na *Figura 19*).

A iluminação é um dos principais problemas quando se trabalha com visão de máquinas (*secção 3.2.2*). Se possível, é importante que esta variável do ambiente esteja controlada. O ambiente onde o Segway está inserido, é um ambiente com elevado volume ocupacional, numa área de 1735m^2, com muitas e diferentes fontes de luz, o que faz aumentar a complexidade do problema (*ANEXO 8*). Para justificar esta afirmação foi construído, com a ajuda de um sensor "LDR", um "raspberryPi" e um medidor de corrente, um equipamento (*Figura 20*) que permite ler as variações de intensidade luminosa que se fazem sentir num determinado local. O esquema de montagem encontra-se disponível no *ANEXO 1.*

O sensor LDR (Resistência Dependente da Luz) aumenta ou diminiu a sua resistência à passagem de corrente consoante a luminosidade que inside na superficie do sensor é menor ou maior, respetivamente. Quanto maior a quantidade de luz que inside sobre o sensor LDR, menor a sua resistência e, pela *lei de Ohm,* maior será a corrente que flui pelo sensor.

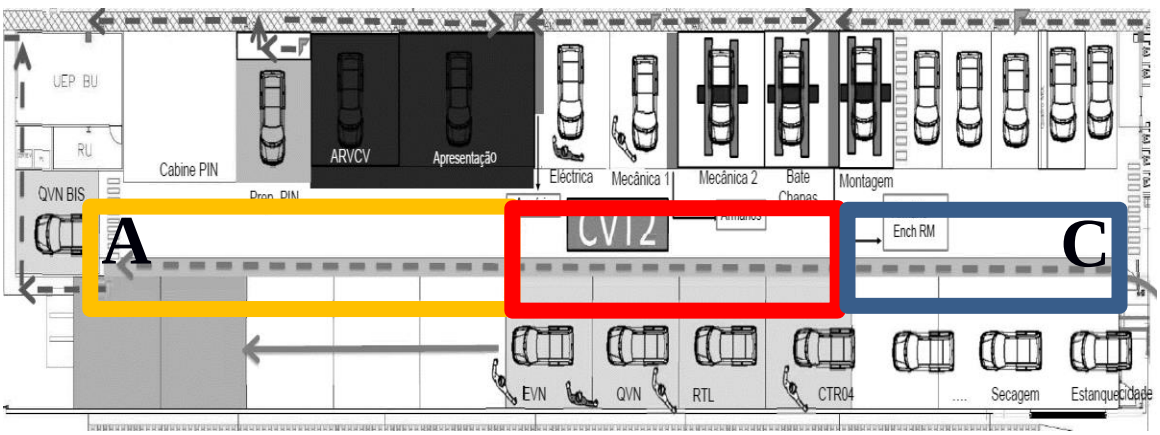


Figura 19 - Mapa 2D da BTU.

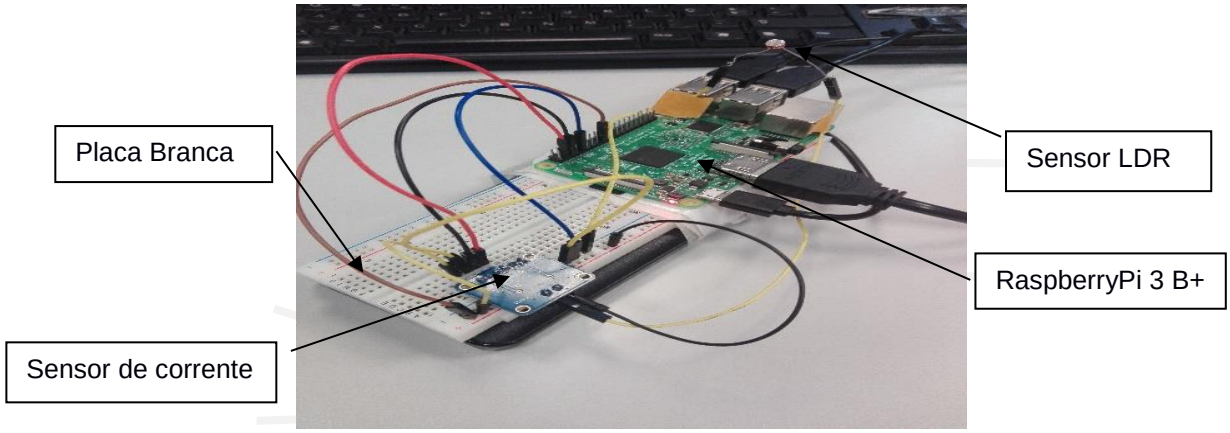


Figura 20 - Medidor de variações de luminosidade.

As fontes de luz, que iluminam a zona da BTU ao longo dos turnos (manhã, tarde e noite), variam do dia para a noite. Durante o dia, e em dias de sol, a iluminação dominante provém do exterior, atingindo o interior pelas clarabóias que estão no teto da fábrica e pelas portas de acesso à fábrica. Em dias nublados ou períodos noturnos, a iluminação dominante passa a ser a iluminação artificial que existe dentro de fábrica. Na *Tabela 6* são sintetizadas as principais fontes de iluminação.

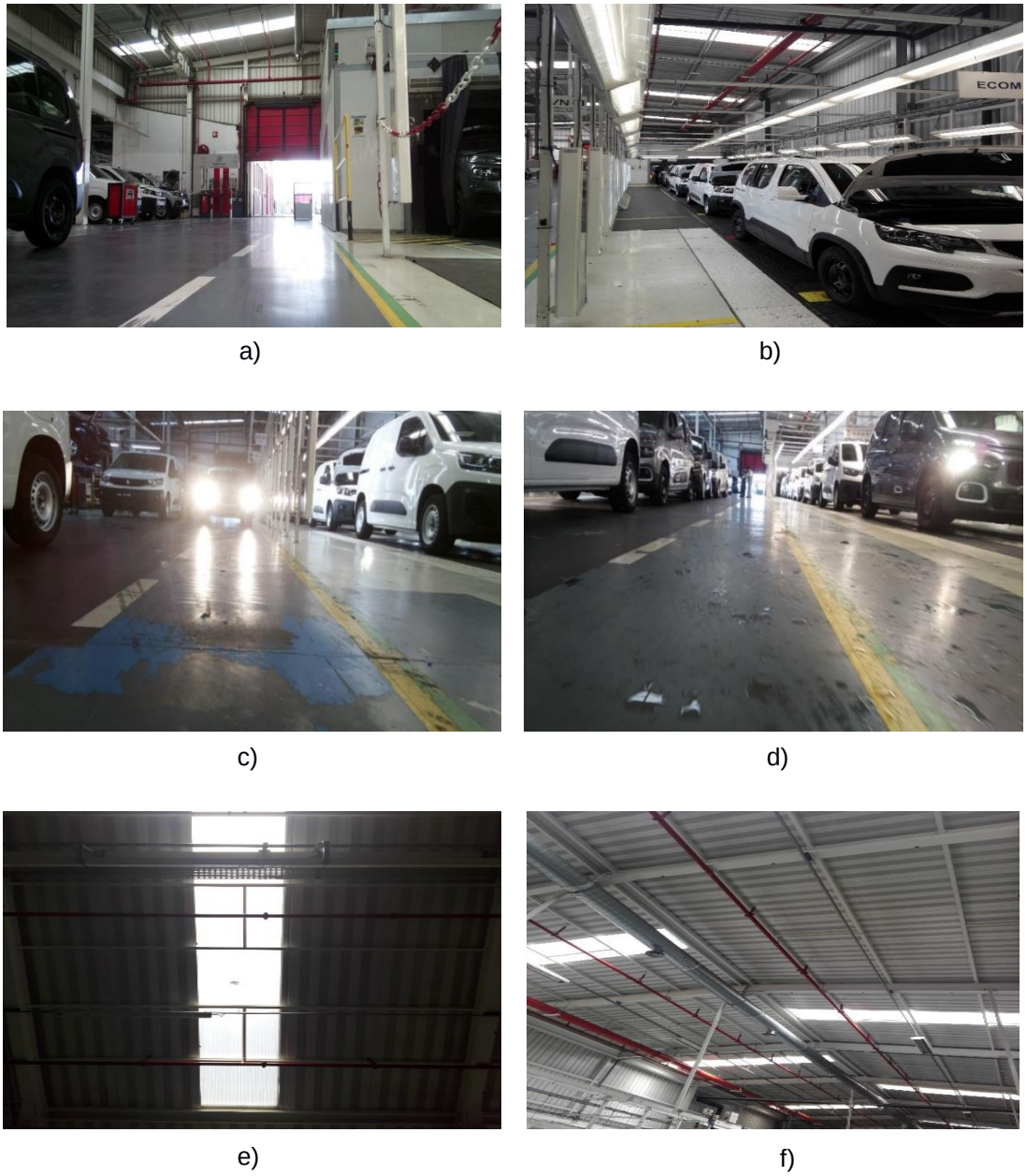
Dia	Clarabóias (Luz Solar) Lâmpadas LED (6500k e 4000k, 3000lm) Portas abertas (Luz Solar)
Noite	Lâmpadas LED (6500k e 4000k, 3000lm) Faróis dos veículos (máximos e médios)

Tabela 6 - Principais fontes de iluminação.

Foi feita uma recolha de várias imagens que se consideram importantes para perceber os problemas que afetam o Segway ao longo do seu percurso autónomo. Em seguida são enumerados essas mesmas situações:

1. Porta para entrada de véiculos, localizada no final do setor C (*Figura 21*-(a)) ;
2. Cerca de 70 lâmpadas LED iluminam o setor B onde é feito o controlo de aspeto dos véiculos (*Figura 21*-(b)) ;
3. Os véiculos que circulam pelo BTU ao longo de todos os setores (A, B e C) têm ligados os faróis (médios e, por vezes, máximos) (*Figura 21*-(c) e (d)) ;
4. Clarabóias no teto da fábrica, espalhas pelos três setores (A, B e C) (*Figura 21*-(e) e (f)) ;
5. A baixa luminosidade no setor C, durante o perídodo da noite (*Figura 22*-(b) e (c)) ;

6. Os raios de sol durante as primeiras horas da manhã são refletidos no piso por onde circula o segway (*Figura 22*-(a)) ;

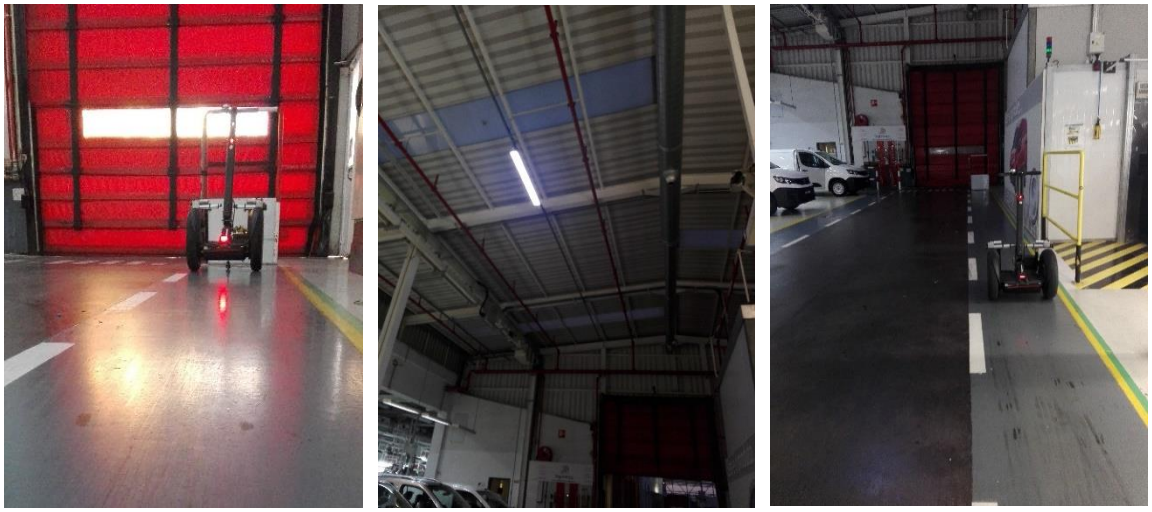


a) b)

c) d)

e) f)

Figura 21 - Diferentes fontes e tipos de iluminação.



a) b) c)

Figura 22 - Diferentes fontes e tipos de iluminação.

Todos estas situações têm um papel importante na facilidade com que o algoritmo consegue ou não detetar a linha pela qual o Segway se guia. Na Figura 23 são ilustradas três situações ((a),(b) e (c)) onde a iluminação existente na fábrica impede que o Segway detete as cores da linha. A consequência disto tudo é que o algoritmo (apresentado na *sub-secção 2.2.3*) não consegue detetar os contornos da linha, ou, se deteta, estes acabam por ter uma forma irregular. Neste caso em

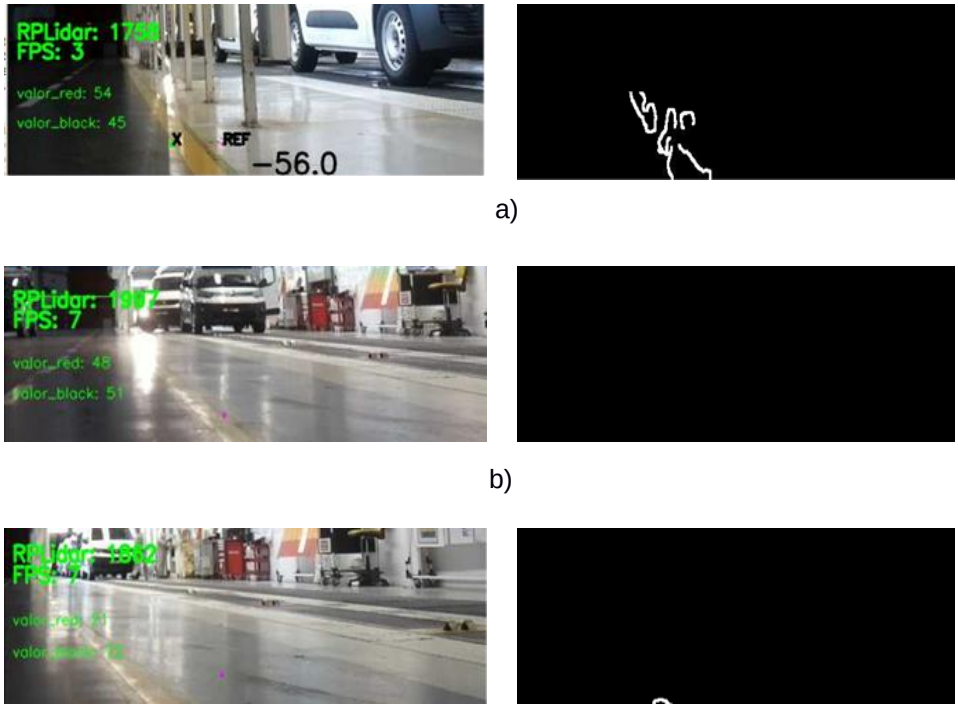


Figura 23 - Efeitos da iluminação na deteção da linha.

concreto, a iluminação que impede o Segway de ver a linha provem do setor B. No entanto, algo semelhante acontece no setor C e no setor A, quando há a passagem de veículos com os faróis ligados ou as portas se abrem para a passagem de veículos.

Como podemos ver nas figuras anteriores (*Figura 21* e *Figura 22*), a variedade de luzes e os ângulos de onde a luz provém é ampla. O excesso de reflexos acaba por ter um impacto significativo no processamento das imagens, uma vez que reduzem significativamente o contraste entre a linha e o meio que o rodeia, tornando difícil, se não impossível, a segmentação da linha.

Aplicando as práticas que se estudaram na *secção 3.2.2,* foram propostas duas soluções para tentar eliminar o impacto que a iluminação tem sobre a eficácia na segmentação da linha. Como não podemos mexer na iluminação exterior, podemos focar-nos em algo menos complexo e que pode ser eficaz no combate ao problema da iluminação, controlar a quantidade de luz que chega ao sensor da câmara *(PiCamera V2).* Propõe-se assim a aplicação de duas técnicas:

- Uso de um filtro Polarizador e UV
- Variação do ângulo da câmara em relação ao piso

Estas duas soluções são propostas com vista a diminuir a exposição da câmara à luz ambiente e, consequentemente, conseguir-se aumentar o contraste entre a linha e o meio.

Os filtros polarizadores são muito usados na fotografia para diminuir os reflexos e excesso de luminosidade, que atinge o sensor da câmara no momento de captura da foto. O filtro, como o nome indica, permite filtrar a radiação de um determinado comprimento de onda, eliminando grande parte da luz que por ele passa. Na Figura *24* podemos ver as diferenças entre uma imagem capturada por uma câmara com filtro polarizador e outra sem filtro polarizador [20] e perceber que as diferenças são significativas.

Figura 24 - Diferenças entre usar um filtro e não usar um filtro [20].

A alteração da posição da câmara em relação ao solo poderá ser uma solução complementar ao uso do filtro polarizador, pois a quantidade de luz que entra no sensor será menor quando maior a sua altura em relação ao solo e quanto maior for o seu ângulo de inclinação.

[20] Sara Leblanc, https://mott.pe/noticias/todo-lo-que-necesitas-saber-acerca-del-filtro-polarizador/

A câmara encontra-se fixa (*Figura 25*-(a) e (b)) sem qualquer grau de liberdade e descentrada em relação à posição central do Segway (*Figura 25*-(b)). Para poder alterar a posição (altura e inclinação) da câmara em relação ao solo foi construído um suporte numa impressora 3D (*Figura 26-(*a) e (b)). Depois de algumas modificações na caixa, este suporte foi fixo à parte superior da caixa negra como se ilustra na Figura *26* – (c) e (d). O sistema passa então a ter um grau de liberdade (ângulo de inclinação do suporte) para se testar qual a melhor posição que leva à situação de maiores contrastes.

a) b)

Figura 25 - (a) Posicionamento fixo da Câmara. (b) Posição da câmara na caixa negra.

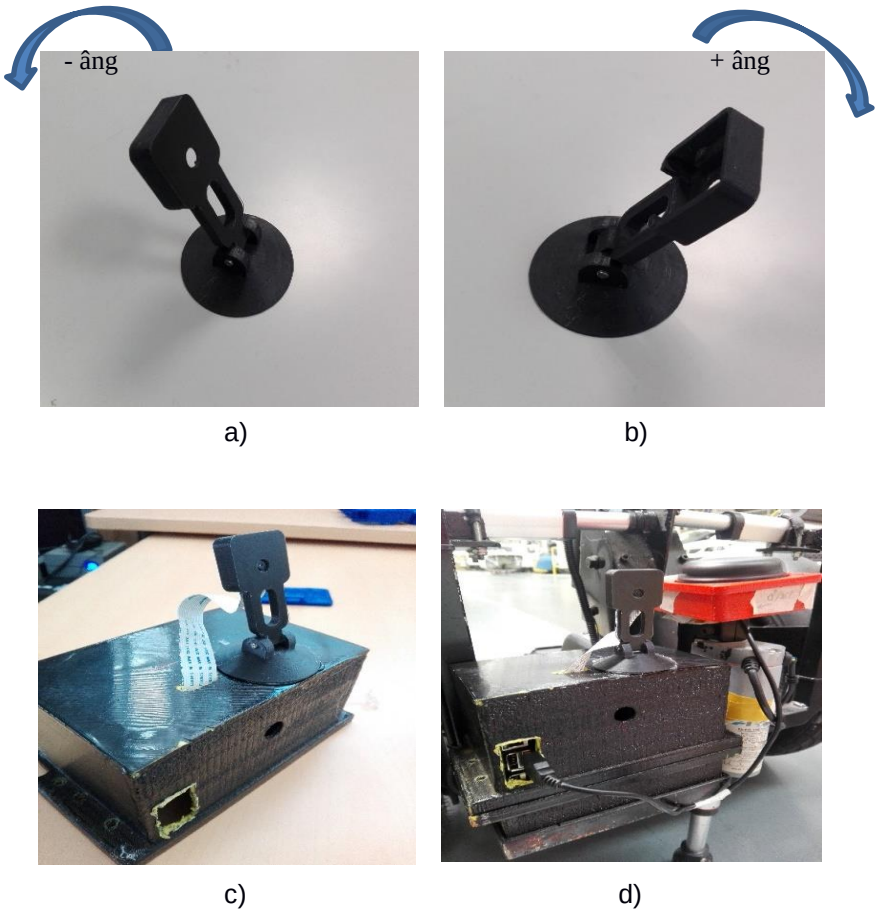


Figura 26 - (a) e (b) Suporte para a câmara. (c) e (d) Disposição final do suporte + câmara.

Mas nem sempre o excesso de luz é o problema principal. Existem momentos, durante o período noturno, em que a baixa luminosidade e as sombras podem dificultar a deteção da linha. Para contornar este problema propôs-se a implementação de um holofote LED (*Figura 27*-(a)) onde as características principais estão na Tabela *7*. Para fazer os ensaios e perceber o respetivo impacto no desempenho do Segway o holofote [21] foi acoplado ao Segway como se mostra na Figura *27*-(b) e (c). Como não se podia prever os impactos finais, sem antes efetuar os respetivos ensaios, o holofote foi fixo ao suporte metálico do Segway com recurso a fita adesiva e alimentado por um cabo de alimentação com cerca de 50 metros.

[21] FalconEye, “FalconEyes”, http://www.falconeyes.com.hk/Product.aspx?id=1516

a)

b)

c)

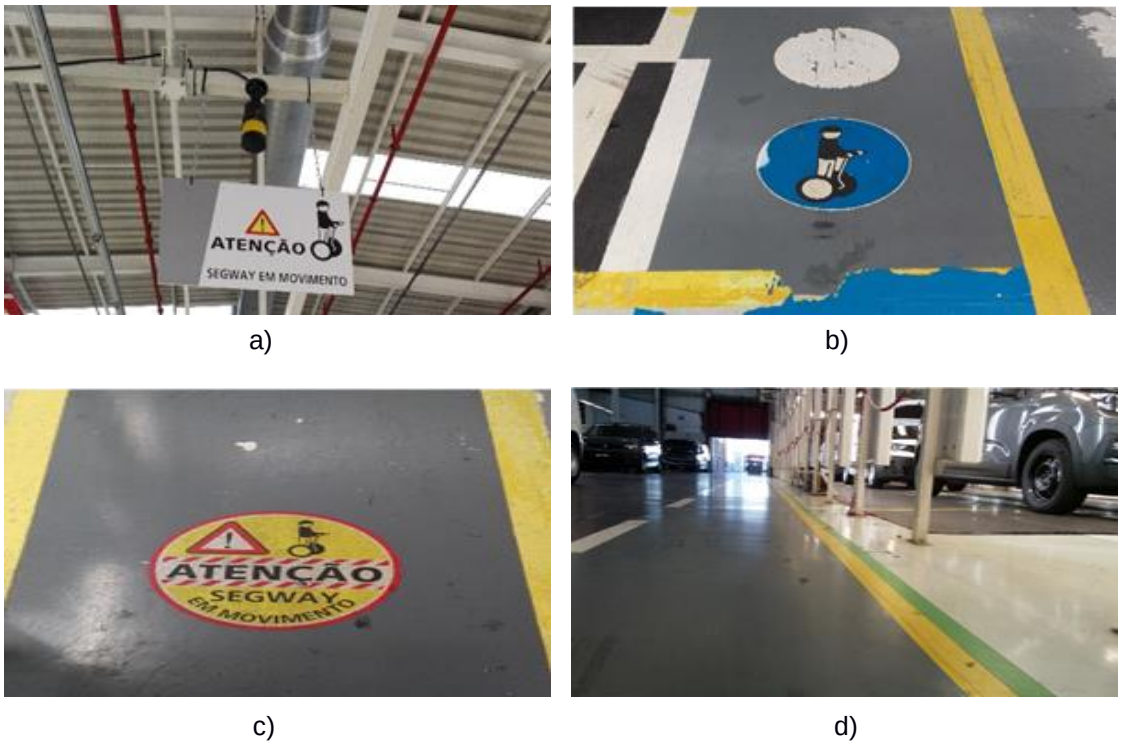
Figura 27 - (a) Falcon Eye DV-216VC[21]. (b) e (c) Ensaios com o holofote.

Número de LEDS (und)	216
Temperatura da Cor (K)	3000 – 7000
Alimentação DC (V)	7 – 12
Dimensões (mm)	180x115
CRI	95
Consumo de energia (W)	13
Alcance da luz (m)	0.5 - 2

Tabela 7 - Caracterização do Holofote Falcon Eye DV-216VC [21]

4.2. CARACTERIZAÇÃO DO PAVIMENTO E LINHA GUIA

O Segway percorre cerca de 70 metros ao longo de um percurso em linha reta, que está devidamente assinalado por placas e marcações no solo (*Figura 28*). Na Figura *28*-(a) ilustra-se uma placa de "atenção". Na Figura *28*-(b), (c) e (d) ilustram-se as marcações feitas no solo (Sinais e Linhas (amarela e verde)).



a) b)

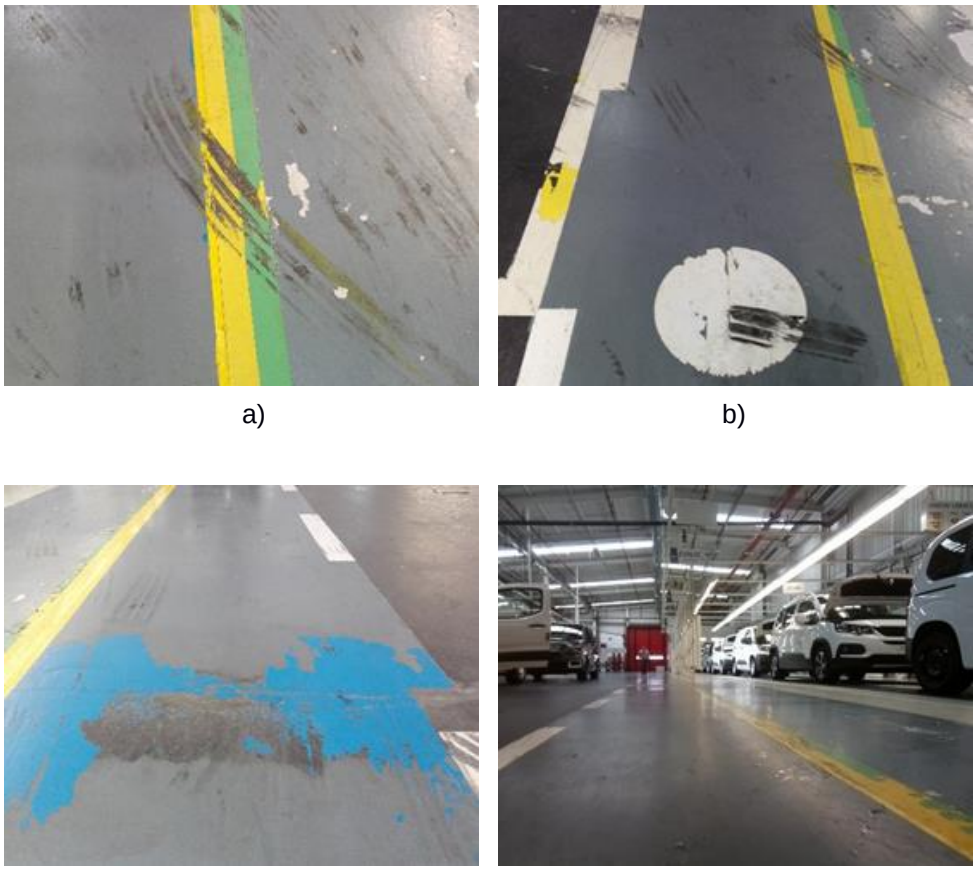
c) d)

Figura 28 - Marcas de sinalização ao longo do percurso do Segway.

Uma das principais características do pavimento que chama a atenção é o facto de o piso não ser regular e apresentar algumas marcas de degradação. A frequente passagem de veículos pela zona onde passa o Segway leva a que haja um crescente desgaste e danificação da linha pela qual o Segway se guia. Na Figura 29-(a) podemos ver marcas de pneus; na Figura 29-(b) podemos observar um pedaço de linha amarela sobre a linha tracejada branca; na Figura 29-(c) temos um pequeno desnível no piso (o piso varia em altura) que se encontra entre o setor A e o setor B; na Figura 29-(d) podemos verificar que as linhas verdes e amarelas se encontram desgastadas. Na mesma figura (*Figura 29*-(d)), percebe-se que o piso por onde o Segway se desloca, facilmente reflete a luz, que sobre ele inside.

Na Figura 30, podemos analisar que o mau estado da linha e a existência de humidade (este fator agrava-se durante os dias de chuva) têm um papel importante na segmentação da linha. Isto é, o algoritmo consegue segmentar a linha pela cor, mas as inúmeras irregularidades que a

linha tem, agravado pelo efeito da água (humidade) que se encontra espalhada pelo chão, fazem com que os contornos detetados na imagem intitulada *"binarizada"* da Figura 30 sejam tudo menos semelhantes aos contornos de uma linha reta no seu estado normal (limpa e sem degradação).



a) b)

c) d)

Figura 29 - Estado do piso ao longo do percurso.

.

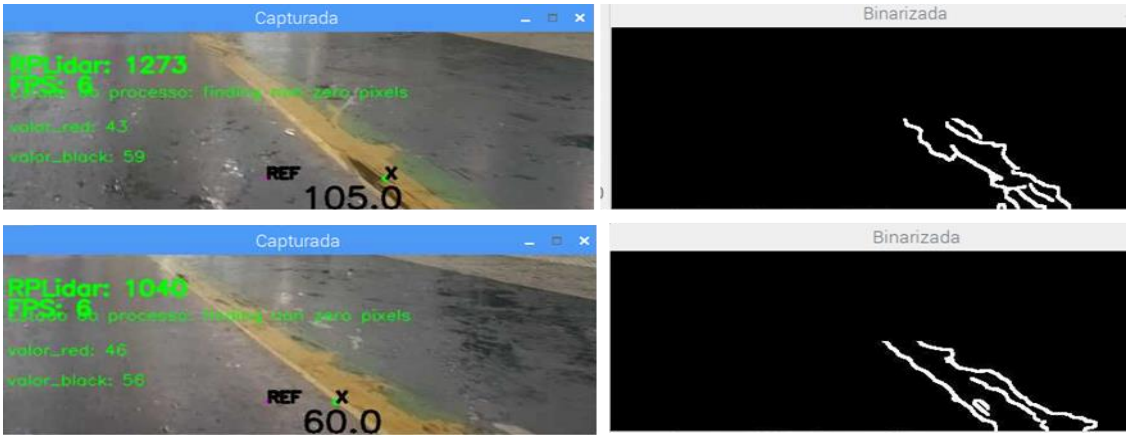


Figura 30 - Impacto do mau estado do pavimento e das linhas que marcam o solo no algoritmo.

Para contornar o problema do mau estado do piso, diminuindo os atritos entre a extremidade do atuador vertical e o piso, e conseguir algum amortecimento nas zonas onde o piso não é regular, propõe-se a substituição do castor que estava até ao momento a ser utilizado (*sub-secção 2.2.2*) por um castor com mola [22](*Figura 31*).

Foi feito um estudo da carga (*Figura 32*) para determinar quais os parâmetros mínimos da mola, para que esta aguente a carga máxima a que o atuador vertical está sujeito. Segundo os dados da *secção 2.2.1,* o Segway não pesa mais de 40Kg e o atuador vertical está ligeiramente posicionado à frente do Segway (*Figura 32*). Logo, podemos concluir, que a força a que o atuador vertical está sujeito é inferior ao valor da força total exercida pelo peso do Segway (*Equação 5*). Por esta razão e pela *equação 4,* conclui-se que a força na extremidade do atuador não ultrapassa os 370N (*Equação 6*). Procurando em alguns catálogos, encontrou-se um castor com mola (com carga máxima de 86Kg, equivalente a 860N), o que é suficiente para amortecer os impactos. Para além desta característica, estes castores possuem um vedante, o que impede a acomulação de resíduos, diminuindo assim o aumento dos atritos a longo prazo. Este equipamento é revestido por aço galvanizado [22] (*Aço 1.3541 / 1.4034*) resistente à corrosão, fator importante face às condições (humidade no piso) em que o Segway opera.

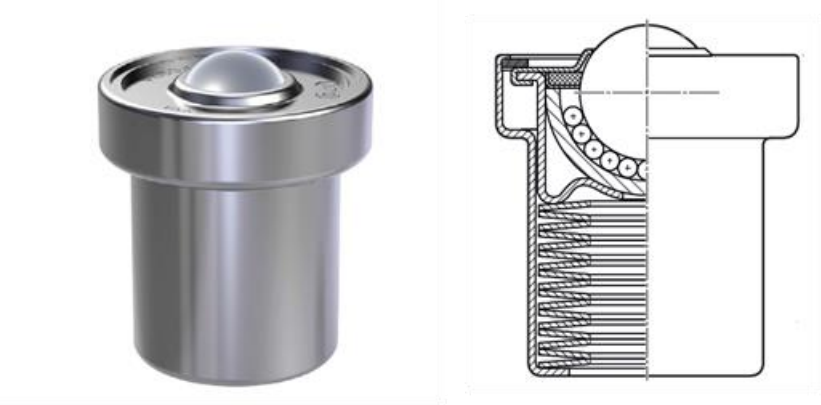


Figura 31 - Castor com mola. Referência: *R0532 222 10 KUF-C22-TF-MFG* [22]

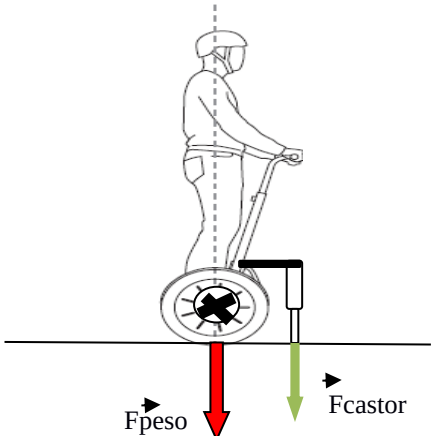


Figura 32 - Representação gráfica das forças aplicadas no Segway.

[22] Group RexRoth Boch, “Ball Transfer Units Catalog”, Catálogo

$$Fpeso = massaSegway * aceleraçãoGravidade \quad (4)$$

$$Fcastor < Fpeso \quad (5)$$

$$Fcastor < 370\ N \quad (6)$$

Para eliminar o mau estado da linha foi adquirido uma nova fita [23] marcadora de solo de cor amarela (*Figura* 33). A linha verde que marca o solo deixa de existir e optou-se por fazer a segmentação de uma só cor (a cor amarela).

Figura 33 - Fita para marcação do solo, disponível em várias cores [23]

[23] tesa® Professional, Catálogo geral tesa® Profissional

4.3. ALGORITMO PARA DETEÇÃO DAS LINHAS

Perante os problemas que se encontraram nas *secções 4.1 e 4.2* e para os quais já se tem uma solução, o algoritmo apresentado na *sub-secção 2.2.3* não demonstra ser o mais robusto possível. Por esta razão, na presente secção são abordados os motivos que levaram à construção de um novo algoritmo. É ainda apresentado e explicado o algoritmo desenvolvido (*sub-secção 2.2.4*) durante o estágio para detetar as linhas que guiarão o Segway ao longo do seu percurso normal.

4.3.1. CARACTERIZAÇÃO DO ALGORITMO ANTERIOR (DEFEITOS)

Mesmo que se consigam eliminar os problemas de iluminação e das condições do piso, o algoritmo (*sub-secção 2.2.3*), que o Segway possui para detetar as linhas, não é o mais robusto. Isto, porque, durante alguns testes feitos para perceber o seu modo de funcionamento, o Segway apresentou as seguintes falhas:

- Desvio da sua rota normal quando aberta a porta do setor C (*Figura 36*) ;
- Desvio da sua rota normal quando, pelo seu lado esquerdo, passavam ou estavam estacionadas carrinhas de cor amarela (*Figura 34*) ;
- Desvio da sua rota normal quando, no campo de visão da câmara, era apanhado um colete ou objeto de cor verde/amarela (*Figura 35*) ;

Durante estes testes, houve a oportunidade de capturar alguns momentos em que se demonstram estas situações. Na Figura *36*, temos uma sequência de fotos tiradas no exato momento em que o Segway perde o seu controlo e se desvia do seu percurso normal. Isto aconteceu devido aos reflexos provenientes de luzes exteriores que se faziam sentir no piso, induzindo o algoritmo em erro. O algoritmo deteta os píxeis de cor amarela (as lâmpadas exteriores são lâmpadas que emitem uma luz quente na ordem dos 3500 K (*Kelvin*), e, por isso, com tonalidade amarela). Como podemos ver, na Figura *36*, imagem capturada já no final do dia, as luzes artificiais da fábrica acabam por ter um maior impacto, como já tinha sido observado na *secção 4.1*. Algo semelhante acontece quando os veículos de cor amarela passam sobre o Segway e refletem a sua cor no piso (*Figura 34*), induzindo o algoritmo em erro, fazendo com que assuma uma trajetória imprevisível.

Os coletes utilizados pelos operadores (*Figura 35*) também são um problema, pois a tonalidade e saturação da sua cor, encaixam dentro dos limites superiores e inferiores utilizados para segmentar a linha no formato HSV definido, o que leva à segmentação dos contornos do colete e induz o algoritmo do Segway em erro, pensando que está a detetar a linha, quando, pelo contrário, está a detetar um objeto que não interessa.

Na Figura *37* podemos ver o motivo pelo qual o segway toma uma trajetória sempre que é detetado um píxel de cor amarela na imagem (a origem do píxel pode ser um reflexo de um objeto de cor amarela, um reflexo de uma luz mais quente ou mesmo um objeto amarelo que esteja dentro do campo de visão da câmara).

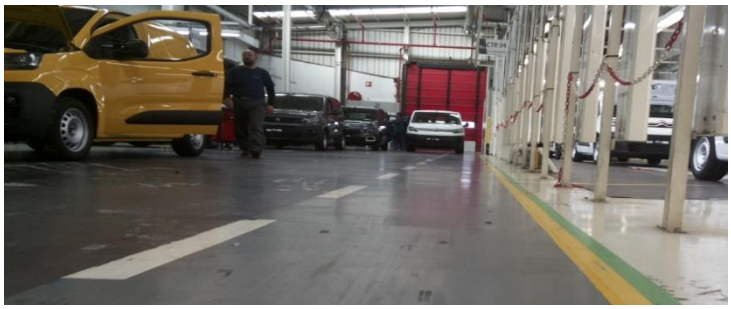


Figura 34 - Veículo amarelo estacionado no lado esquerdo do percurso.

Figura 35 - Coletes amarelos utilizados pelos operadores.

Figura 36 - Sequência de imagens em que o Segway se perde.

Figura 37 - Efeito dos reflexos amarelos no algoritmo.

4.3.2. CARACTERIZAÇÃO DO ALGORITMO DESENVOLVIDO

Para colmatar as falhas apresentadas pelo algoritmo apresentado na *secção 2.2.3*, foi desenvolvido um novo algoritmo capaz de ultrapassar os problemas identificados. Um algoritmo imune aos reflexos e capaz de ignorar situações momentâneas (sombras, humidade no piso, encadeamentos da câmara devido aos faróis dos veículos) sem que o seu desempenho seja influênciado foi o objetivo a ter em conta. Depois de feita uma leitura aos vários casos práticos e dicas teóricas dadas por alguns experientes na área (referenciados no capítulo três), foi estruturado um plano para desenvolver o novo algoritmo.

O algoritmo desenvolvido passa por segmentar a linha amarela (lado direito na *Figura 38*) e a linha a tracejado branca (lado esquerdo na *Figura 38*), com base em algumas das suas características (cor, inclinação, comprimento). Tomou-se a decisão de segmentar estas duas linhas, pois todo o percurso que o Segway percorre é delimitado pelas mesmas. O objetivo final deste algoritmo é permitir que o Segway reconheça os limites (linha amarela e linha branca), como a única área/região, por onde ele se deve movimentar.

Figura 38 - Imagem exemplo para o algoritmo desenvolvido.

Antes de se começar a desenvolver o novo algoritmo, o pavimento foi marcado, do lado direito, com uma nova fita de cor amarela.

Com recurso à Picamera V2, que está acopolada no Segway, foram tiradas centenas de fotos em circuntâncias diferentes ao longo dos três turnos (Turno da manhã, tarde e noite). Estas fotos captaram momentos normais (sem grandes interferências do meio exterior), mas também

momentos em que o algoritmo tinha de ser robusto o suficiente para não ser induzido em erro. Destes últimos momentos captados destacam-se:

- Momentos em que os veículos amarelos se encontram estacionados no lado direito da pista.
- Momentos em que objetos de cor amarela (coletes refletores, sapatilhas, entre outros) se encontram sobre a pista.
- Momentos em que os operadores obstruem parcialmente a linha.
- Momentos em que os faróis dos carros refletem sobre o piso a sua luz.

São estes pormenores que podem tornar o novo algoritmo melhor, comparado com o algoritmo anteriormente utilizado. As fotos foram capturadas com uma resolução de 649 (comprimento) por 480 (altura) píxeis. O sistema de coordenadas de cada imagem está representado na Figura 39. Como podemos ver, na Figura 39, a origem encontra-se no canto superior esquerdo. Todos os cálculos que são elaborados neste secção baseiam-se neste sistema de coordenadas.

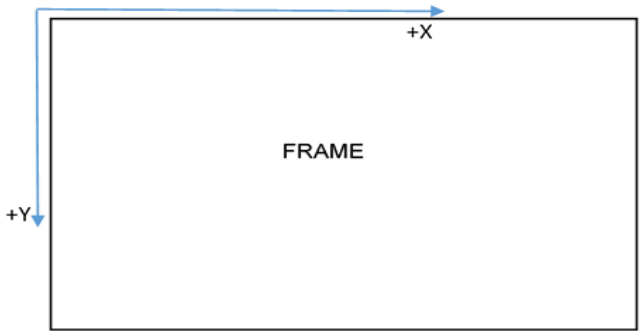


Figura 39 - Sistema de coordenadas cartesianas.

Passamos agora à explicação do algoritmo, que se encontra disponível no *ANEXO 3.* O algoritmo divide-se em duas funções. A primeira função, faz a segmentação da linha a tracejado branca e, a segunda função, faz a segmentação da linha contínua amarela. Começamos por explicar a deteção da linha branca a tracejado.

As frames capturadas pela Picamera V2 vêm num formato RGB. No entanto, optou-se por analisar a imagem em tons cinza (preto e branco) (*Figura 41*-(a)), uma vez que a linha a detetar, é de cor branca.

Em seguida, aplica-se um filtro *Canny (secção 3.2.1.)* com o objetivo de detetar os contornos da linha branca tracejada na imagem (*Figura 41*-(a)). O filtro Canny é aplicado na imagem da Figura *41*-(a) e deteta os contornos da linha branca tracejada (*Figura 41*-(b)). Contudo, outros contornos também são detetados. Como muita da informação que está na imagem não é útil, nem interessa para segmentar a linha branca, a imagem é cortada, definindo-se assim a região de interesse na

Figura *41*-(c). Na Figura *42*, está representada mais uma vez, a região de interesse definida para detetar a linha branca tracejada, mas, desta vez, a cores para se entender melhor.

Na fábrica existem muitos mais objetos, com cor e geometria, semelhantes à linha branca tracejada. Estes objetos podem facilmente conduzir o algoritmo a medições finais erradas. Ao definir a região de interesse elimina-se a probabilidade de serem detetadas contornos que não pertencem à linha branca tracejada.

Em seguida, é aplicado um filtro atenuador de ruído (*smoothing)* sobre os contornos detetados de modo a torná-los mais suaves (*Figura 41*-(d)). É na imagem da Figura *41*-(d) que se aplica a função para detetar segmentos de reta, o *HoughLinesP.* Esta e as funções anteriormente utilizadas, estão disponíveis na biblioteca online *OpenCV.*

A função *HoughLinesP* é uma versão mais otimizada da função *HoughLines* [14], que retorna as coordenadas das extremidades dos segmentos de reta encontrados. Analisando a Figura *41*-(c), podemos verificar, que os contornos, que definem a linha branca são maiores em comprimento, que o resto dos contornos detetados. Após a definição da região de interesse, o comprimento das linhas a detetar é o principal parâmetro, que permite eliminar a maioria das linhas, que não têm qualquer informação útil e que a função *HoughLinesP* permite que se defina como parâmetro de entrada. Na Figura *40* pode-se ver as coordenadas dos pontos extremos de vários segmentos de reta (*i-1, i* e *i+1*), que a função *HoughLinesP* devolve quando aplicada numa imagem. Estes valores são colocados numa matriz coluna para serem utilizados em cáculos posteriores (pós-processamento).

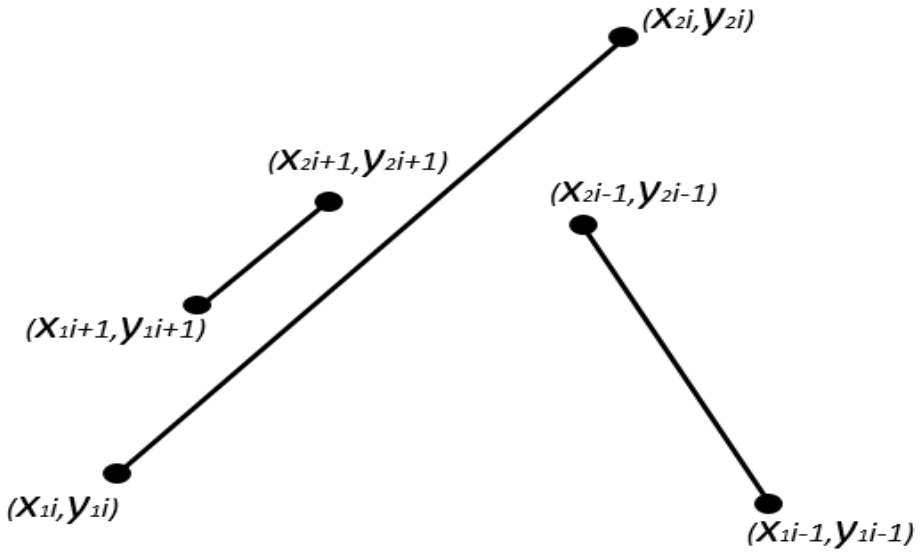
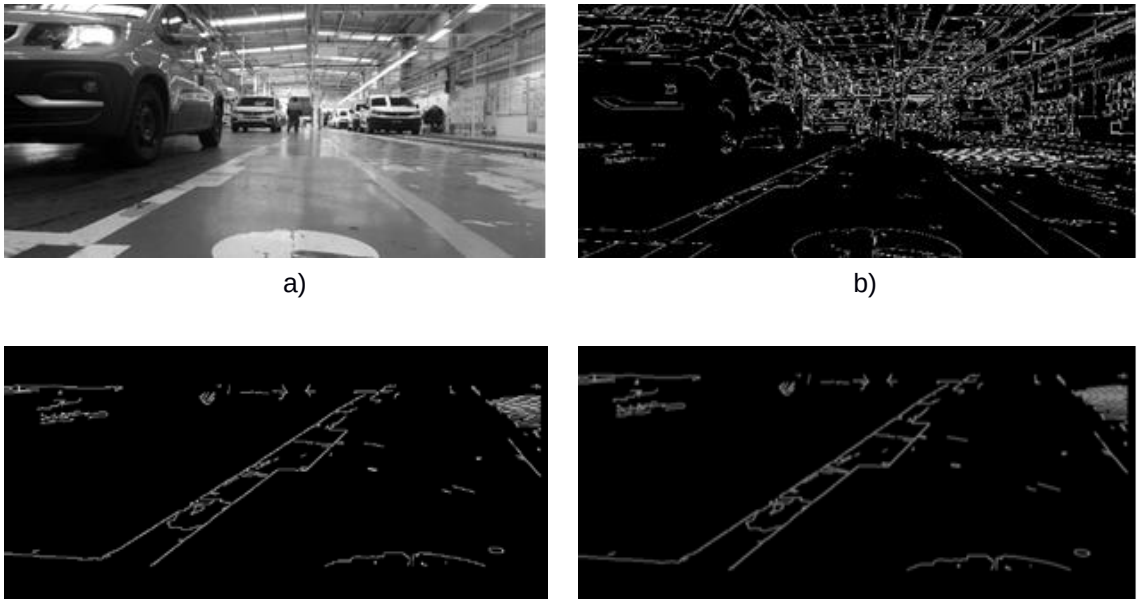


Figura 40 - Valores devolvidos pela função *HoughLinesP*



a) b)

c) d)

Figura 41 - Segmentação da linha branca tracejada.

Figura 42 - Região de interesse para detetar a linha branca tracejada.

Hmin	**Smin**	**Vmin**
96	112	60
Hmax	**Smax**	**Vmax**
120	255	255

Tabela 8 - Valores HSV para segmentar a cor amarela.

Para a segmentação da linha amarela contínua, o princípio é o mesmo que na linha branca descontínua, mas há pequenas diferenças que se devem destacar. A começar pelo formato da imagem. Agora, em vez da imagem ser convertida para um formato de tons cinza, ela é transformada para um formato HSV. É neste formato que se faz uso da principal característica que a linha tem: a cor. Para segmentar a linha no formato HSV foram definidos os valores da Tabela 8. Com estes valores conseguimos transformar a imagem da Figura 38 na imagem da Figura 43-(a). Em seguida, semelhante ao procedimento usado para a segmentação da linha branca, é aplicado um filtro Canny (Figura 43-(b)) e um filtro smoothing (atenuador de ruído) (Figura 43-(c)). Tendo já os contornos da linha segmentada, é importante que outros contornos não interfiram no restante processamento. Por isso, é criada na imagem, uma região de interesse, com um simples corte das regiões na imagem, que não interessam (Figura 44). Depois disto, aplica-se a função HoughLinesP, que devolve as coordenadas em píxeis das extremidades dos segmentos de reta, que mais se aproximam com os contornos da linha amarela na imagem.

a) b) c)

Figura 43 - Segmentação da linha amarela.

Figura 44 - Definição da região de interesse.

Terminado o processamento das imagens, passamos agora ao pós-processamento. Nas imagens da Figura *41*-(d) e Figura *43*-(c) é aplicada a função *HoughLinesP.* Esta função, disponível na biblioteca online *OpenCV,* recebe como parâmetros de entrada o comprimento mínimo em píxels dos segmentos de reta que se pretende detetar e o intervalo máximo entre pixéis para que estes sejam considerados um segmento de reta (*ANEXO 3*). Analisando de novo as Figuras 41 - (d) e Figura *43* – (c) percebe-se que os contornos da linha branca e da linha amarela respetivamente são neste caso maiores que o resto dos contornos detetados. Definindo um comprimento mínimo através da função HoughLinesP podemos logo à partida eliminar um conjunto de contornos que não interessam para o processamento.

Para uma dada reta i é devolvido pela função *HoughLinesP* o valor das coordenas das suas extreminades (*X1i,Y1i*) e (*X2i,Y2i*). Aplicando a equação:

$$declive = \frac{y2-y1}{x2-x1} \quad , \text{ com } x2 - x1 \neq 0, \tag{7}$$

conseguimos determinar o declive para cada segmento de reta que foi encontrado na imagem. Depois de se separar pelo seu comprimento, os contornos encontrados, podemos ainda separá-los pelo seu declive.

Assumindo que o Segway deverá andar sempre dentro do limite definido pelas duas linhas, a linha branca e a linha amarela apresentam declives contrários. Ou seja, a linha branca apresenta um valor de *declive* negativo e a linha amarela apresenta um valor de declive positivo. Depois de alguns testes empíricos, os valores de *declive* foram afinados tendo-se usado os seguintes valores como referência (*Equação 8)*, para separar os contornos, que pertenciam ou não às linhas a segmentar:

$$\begin{cases} declive < -0.4\,, & para\ detetar\ linha\ branca \\ declive > 0.6\,, & para\ detetar\ linha\ amarela \end{cases}. \tag{8}$$

Isoladas os contornos pelo seu declive resta-nos um conjunto de segmentos de retas que na sua maioria são representativos dos contornos das linhas brancas e amarelas a detetar. No entanto, o comprimento das retas e o seu declive não são fatores suficientes para segmentar as duas linhas (branca e amarela). É dada a Figura 45 como exemplo. Na mesma figura vemos três retas com igual declive e igual comprimento, contudo em posições diferentes na imagem.

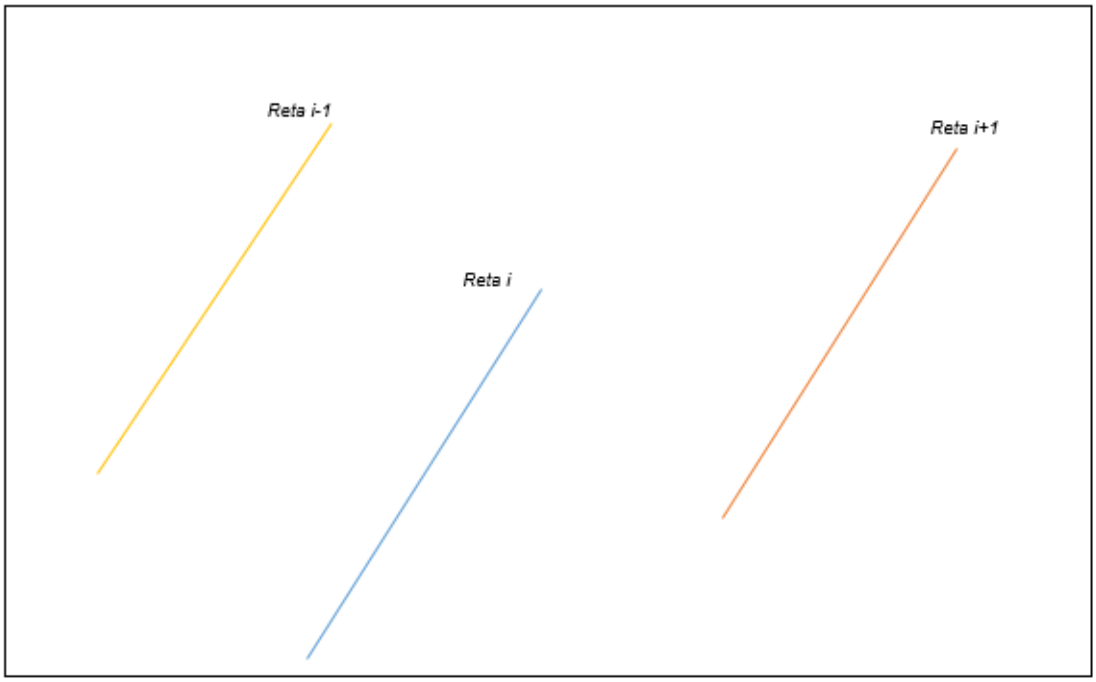


Figura 45 - Segmentos de reta detetados numa imagem.

Para conseguir isolar as retas que pertencem apenas às linhas brancas e amarelas, para cada segmento de reta é calculado o ponto de interseção com o eixo *x = 0* e com o eixo *y = altura da* imagem, na imagem. Para determinar o ponto de interseção de um dado segmento de reta *j f*oi utilizada a equação que definie uma reta, dada por

$$y = m \times x + b\,, \tag{9}$$

Onde b é a interseção da reta com o eixo x = 0, m o declive e (x,y) um ponto qualquer pertencente à reta. Da função HoughLinesP sabemos os pontos (X1j,Y1j) e (X2j,Y2j) para cada segmento de reta. Com recurso ao sistema

$$\begin{cases} y1j = m \times x1j + b \\ y2j = m \times x2j + b' \end{cases} \tag{10}$$

calcula-se os parâmetros m_j e b_j para cada segmento de reta j. Depois de saber estes parâmetros facilmente se retiram os pontos de interseção (*interseçãox, interseçãoy*), com x=0 e y=altura da imagem para cada segmento de reta j pelas equações

$$interseçãoy = y - mx, \tag{11}$$

$$\begin{cases} interseçãox = \frac{y-b}{m}\ , & linha\ amarela \\ interseçãox = \frac{-b}{m}\ , & linha\ branca \end{cases} \quad (12)$$

Tendo os valores das interseções para cada segmento de reta j é feito uma média destes valores para encontrar um valor de referência. O valor médio para as interseções em x (*Xaverage*) e o valor médio para as interseções em y (*Yaverage*) é dado pelas equações

$$Xaverage = \frac{\sum_0^{lastLine} InterseçãoX}{Length(InterseçãoX)}\ , \quad (13)$$

$$Yaverage = \frac{\sum_0^{lastLine} b}{Length(b)}. \quad (14)$$

Caso não seja detetada nenhuma reta, os valores são considerados nulos e não entram para os cálculos seguintes. Neste momento os segmentos de reta candidatos para representar a linha branca tracejada têm como valor de referência (cada segmento de reta) um par de coordenadas (Xaverage, Yaverage). O mesmo acontece para os segmentos de reta candidatos a representar a linha amarela contínua. Mas para que estes segmentos de reta sejam considerados válidos e entrem nas contas finais que permitem representar as linhas delimitadoras do percurso é necessário que um último requesito seja cumprido. Ou seja, para cada segmento de reta j com interseções (interseçãox, interseçãoy) é calculado o valor da distância em pixeís entre os valores de interseção do segmento de reta j e o valor médio de referência (Xaverage,Yaverage) (*Equação 15*). Se a distância do segmento de reta j (distância$_j$) for inferior a uma distância mínima de referência, o segmento de reta é considerado válido e entra nos cálculos para determinar a linha representante. Caso contrário não é considerado um contorno válido. O valor da distânciaj para um determinado contorno j é dado pela equação

$$distânciaj = \sqrt{(Xaverage - Interseçãox)^2 + (Yaverage - Interseçãoy)^2}. \quad (15)$$

Na Figura 46*,* os pontos vermelhos representam as retas encontradas pela função *HougLinesP* e que se encontram dentro do valor de declive e comprimento pré-definidos. Cada ponto vermelho tem como coordenadas o ponto de interseção em Y (*interseçãoy*) e em X (*interseçãox*). O triângulo verde, na Figura 46*,* representa o valor médio, que foi calculado nas *equações 13 e 14*, e que serve de referência para determinar se uma reta é válida ou não.

Na Figura 47*,* é representado um gráfico semelhante ao gráfico da Figura 46, só que desta vez é representada com um circulo a azul, a distância de referência, que corresponde à distância

máxima que os segmentos de reta podem ter para serem consideradas válidos e entrar ou não no cálculos para determinar as linhas representativas dos limites do percurso.

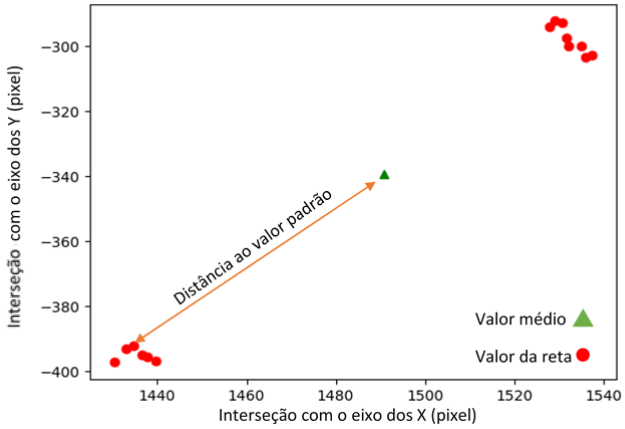


Figura 46 - Segmentos de reta e valor da distância.

Na Figura 47, percebemos à partida que algumas retas foram colocadas de parte pois estão localizados fora do círculo delimitador. Estas retas não entram no cálculo das linhas virtuais que representam a linha amarela e branca.

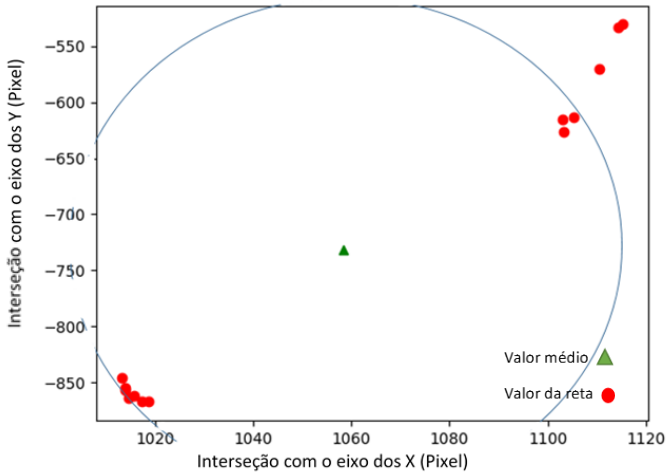


Figura 47 - Definição dos segmentos de reta que entram para o cálculo da linha virtual.

Resumindo, existem três requisitos para que um segmento de reta seja considerado válido e seja usado para o cálculo e ilustração da linha virtual representativa, que são:

1. Estar dentro dos valores de declive e comprimento desejado. Para representar a linha continua amarela têm de ter um declive positivo (>0.6) , e para representar a linha tracejada branca têm de ter um declive negativo (< -0.4).
2. Ter os seus pontos de interseção com os eixos X e Y a uma distância mínima dos valores de referência (valores médios).

Caso não representem nenhum destes parâmetros, os segmentos de reta assumem a coloração avermelhada. Caso cumpram o primeiro parâmetro, assumem a coloração alaranjada. Por fim, se estiverem dentro dos dois parâmetros, assumem a coloração esverdeada e entram no cálculo dos parâmetros para encontrar a linha virtual representativa (*Figura 48*). Na Figura *48* podemos ver representado o processo que leva à escolha das melhores linhas que representam o tracejado branco (*Figura 48*-(a)) e a linha contínua amarela (*Figura 48*-(b)).

a) b)

Figura 48 - Seleção dos segmentos de retas pelo algoritmo.

No final do pós-processamento, antes de desenhar a linha virtual representativa, é aplicado um filtro de ruído (filtro média), à semelhança do que foi abordado na *secção 3.2.1.* Este procedimento permite reduzir os ruídos (sombras, reflexos, objetos ou operadores a obstruir parcialmente a linha, entre outros) que possam aparecer. Ou seja, mesmo que em alguma frame/imagem não seja detetada uma das linhas por algum motivo, a linha virtual é na mesma desenhada sem grandes variações, pois para o seu cálculo usa-se como referência a média dos valores obtidos nas frames anteriores

Quanto maior o número de frames, mais lenta se torna a linha a reagir às mudanças e a calcular os parâmetros da nova linha virtual e correspondente representação gráfica. Quanto menor o número de frames mais rápida é a reação para corrigir e se enquadrar com a nova linha. Porém torna-se mais suscetível ao ruído. É importante encontrar um ponto de equilíbrio. Tendo selecionado os segmentos de retas que melhor representam os limites do percurso, é feito, mais uma vez, a média dos seus valores de *interseçãox* e *interseçãoy e estes* valores médios são usados para representar graficamente as linhas virtuais representativas na imagem (*Figura 51*).

Compara-se em seguida dois gráficos. O primeiro, mostra a variabilidade da interseção da linha virtual representativa da linha contínua amarela no eixo dos X, sem filtro (*Figura 49*). O segundo, mostra a variabilidade da distância entre o ponto de interseção da linha virtual com o eixo X e o centro da imagem no eixo X, com o uso de um filtro média (*Figura 50*).

Podemos verificar, por comparação da Figura *49* e *Figura 50* , que sem o uso de um filtro, para o mesmo intervalo de distância percorrida, temos maiores oscilações e amplitudes de valores do que com o uso do filtro.

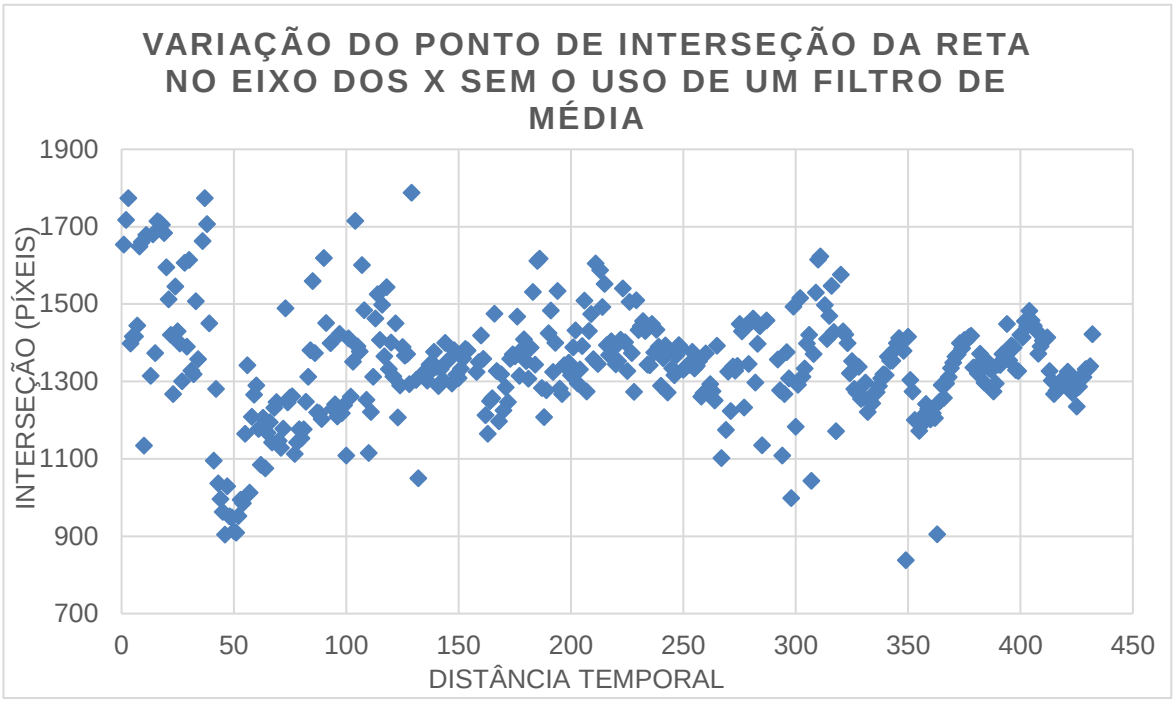


Figura 49 - Variação dos valores sem filtro.

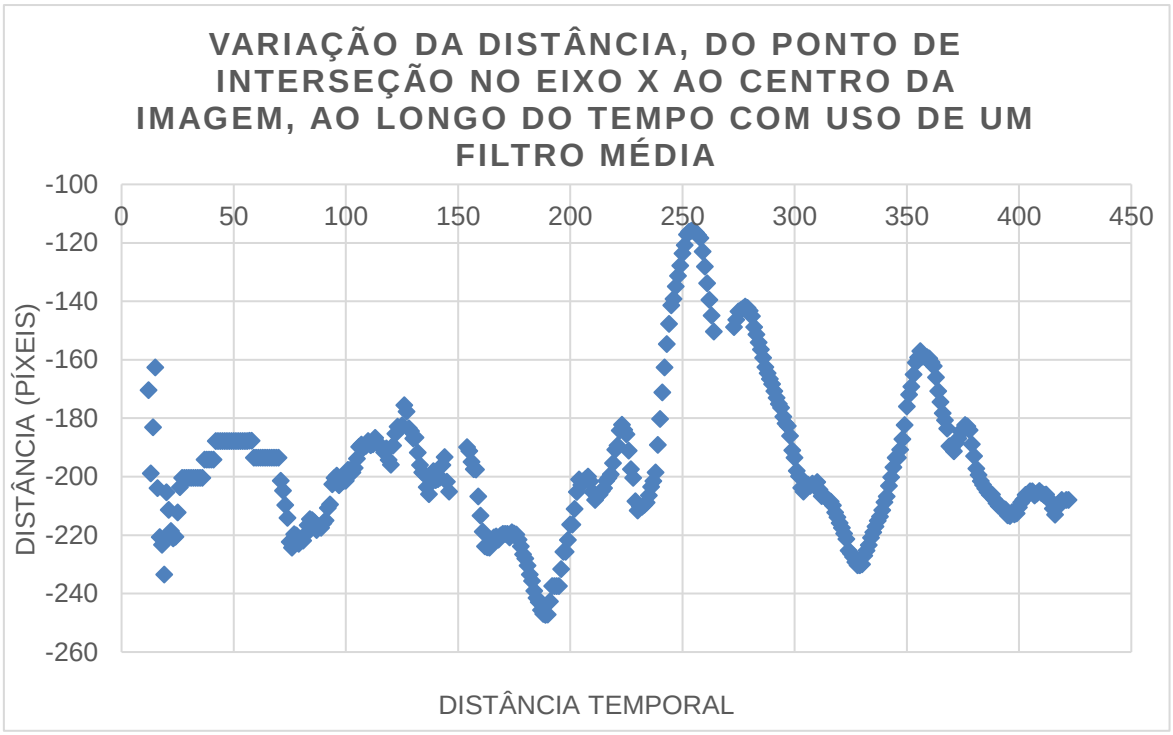


Figura 50 - Variação dos valores com filtro.

Na Figura 51, é ilustrado o resultado final do algoritmo desenvolvido. Temos uma linha virtual a amarelo, que representa o limite direito do percurso, resultado da segmentação da linha amarela contínua, e uma linha virtual a verde, que representa o limite esquerdo do percurso e é resultado da segmentação da linha branca tracejada.

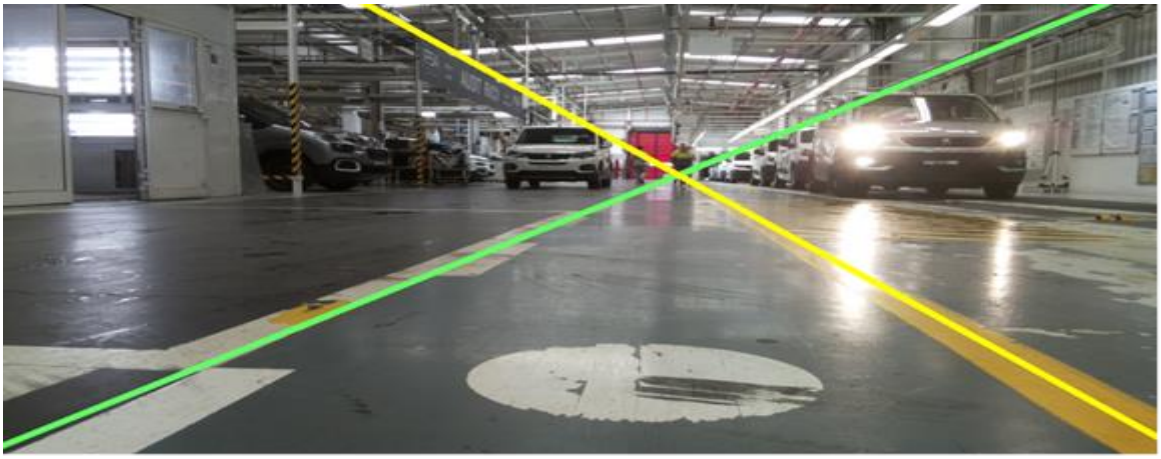


Figura 51 - Linhas virtuais representativas.

4.4. NAVEGAÇÃO E ORIENTAÇÃO

Neste sub-capítulo é apresentado o modo como o Segway se orienta no espaço face aos resultados apresentados no *sub-capítulo 3.3.*

Neste momento, o algoritmo novo é capaz de delimitar os limites do percurso, no entanto, é necessário desenvolver um conjunto de reações para dar resposta aos vários tipos de resultados que o algoritmo apresentado anterioremente possa ter.

No cenário ideal, se o algoritmo detetar ambas as linhas dos dois lados (esquerdo e direito) é dada ordem para o cálculo da área pela equação

$$A = \frac{|InterseçãoEixoXLinhaEsquerda - InterseçãoEixoYLinhaDireita\ | \times Altura}{2}, \quad (16)$$

em píxeis, do triângulo formado pelos pontos de interseção das linhas virtuais (*Figura 52*, ponto 1) e o ponto de interseção de cada linha com o eixo dos x (*Figura 52*, pontos 2 e 3).

Onde

$$Altura = (Ponto\ de\ Interseção\ das\ duas\ linhas - Altura\ da\ imagem). \quad (17)$$

Esta área representa o caminho que o Segway pode percorrer. Posto isto, se esta área for maior ou igual aos valores mínimos, é enviado o menor valor (valor absoluto de P1 ou P2) para o algoritmo de controlo de direção. Caso a área não atinja os valores mínimos, é dada ordem para parar o Segway.

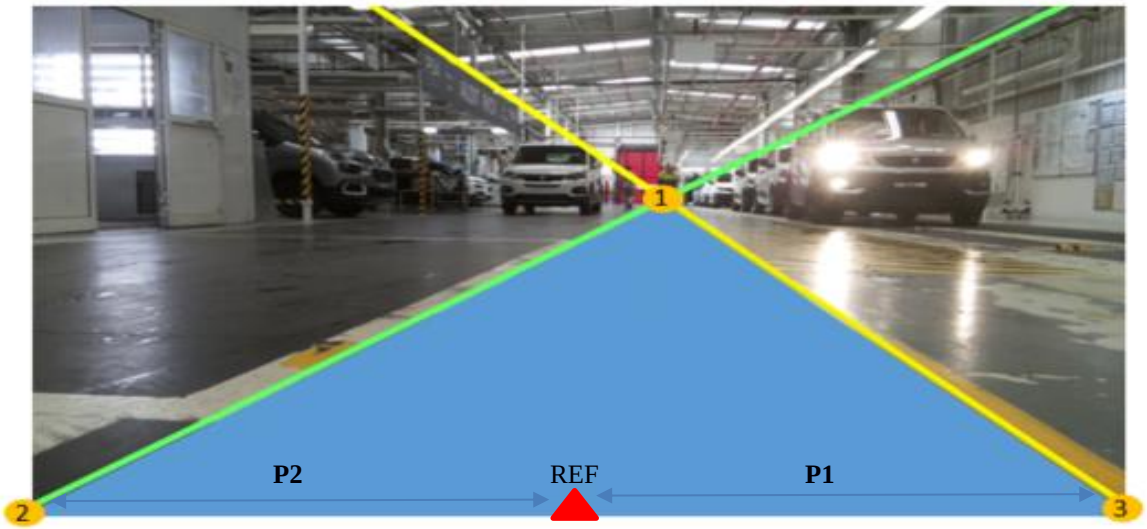


Figura 52 - Representação da área por onde o segway pode andar.

Os pontos P1 e P2 são calculados pelas seguintes equações:

$$\begin{cases} P1 = Ponto3(x) - REF(x) \\ P2 = Ponto2(x) - REF(x) \end{cases}; \quad (18)$$

Na Figura 53, mostra-se o resultado após a aplicação do algoritmo das linhas virtuais. Como foram detetados ambos os limites do percurso, o algoritmo calculou a área que representa o piso, por onde o qual o Segway se pode mover e, como este valor é superior ao valor de referência (valor estipulado pelo operador), foi dada ordem para que o Segway se desloque com a mensagem ("*MOVE ON*").

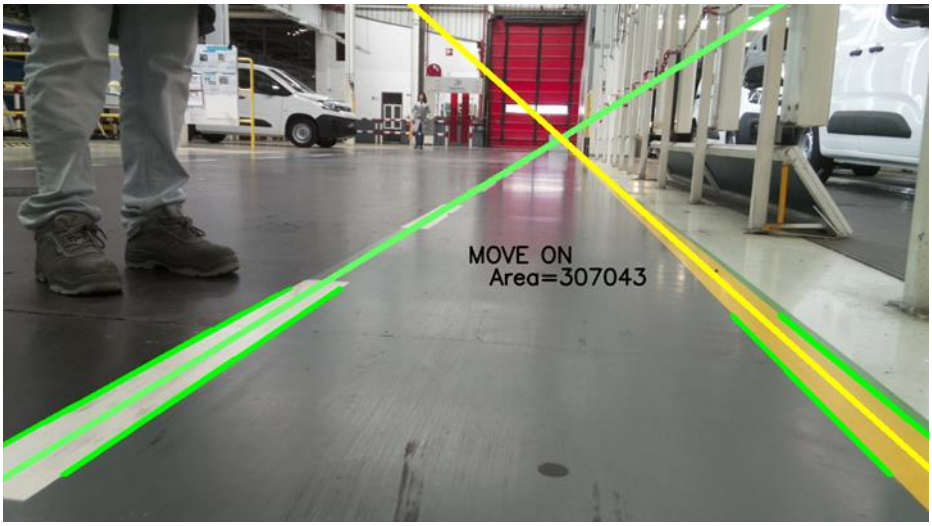


Figura 53 - Situação em que são detetadas ambas as linhas.

Pode acontecer que o algoritmo encontre apenas e uma só linha, o que significa que o Segway pode estar a aproximar-se muito de um dos limites laterais e têm de ser dadas ordens para que ele não saia fora dos limites (*Figura 54*). Na Figura *54*, sempre que o algoritmo não deteta uma das linhas, é mostrada uma mensagem para que o algoritmo se guie apenas pela linha que consegue detetar. A ideia de continuar a andar, mesmo não detetando uma das linhas, surge devido às condições de iluminação, que podem impedir a deteção da linha num dos lados, estando o lado oposto menos afetado. Nestes casos a única opção para o algoritmo do Segway é conseguir traçar a trajetória recorrendo apenas aos dados fornecidos por uma das linhas detetada.

Por último, caso o Segway não visualize qualquer linha, deve imobilizar-se, uma vez que pode estar fora do percurso (*Figura 55*).

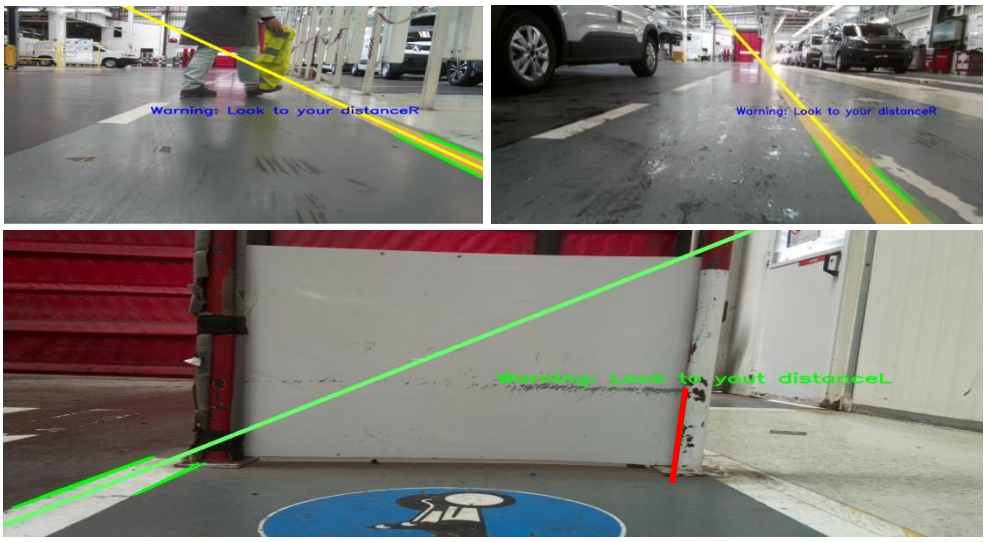


Figura 54 - Situação em que é detetada apenas uma das linhas.

Figura 55 - Não deteta linha.

É face a estes diferentes cenários (*Figura 53,Figura 54* e *Figura 55*), que o algoritmo do Segway define se deve ou não continuar a sua marcha. Na Figura *56* é apresentado um fluxograma que resume todas as ações a desempenhar pelo Segway.

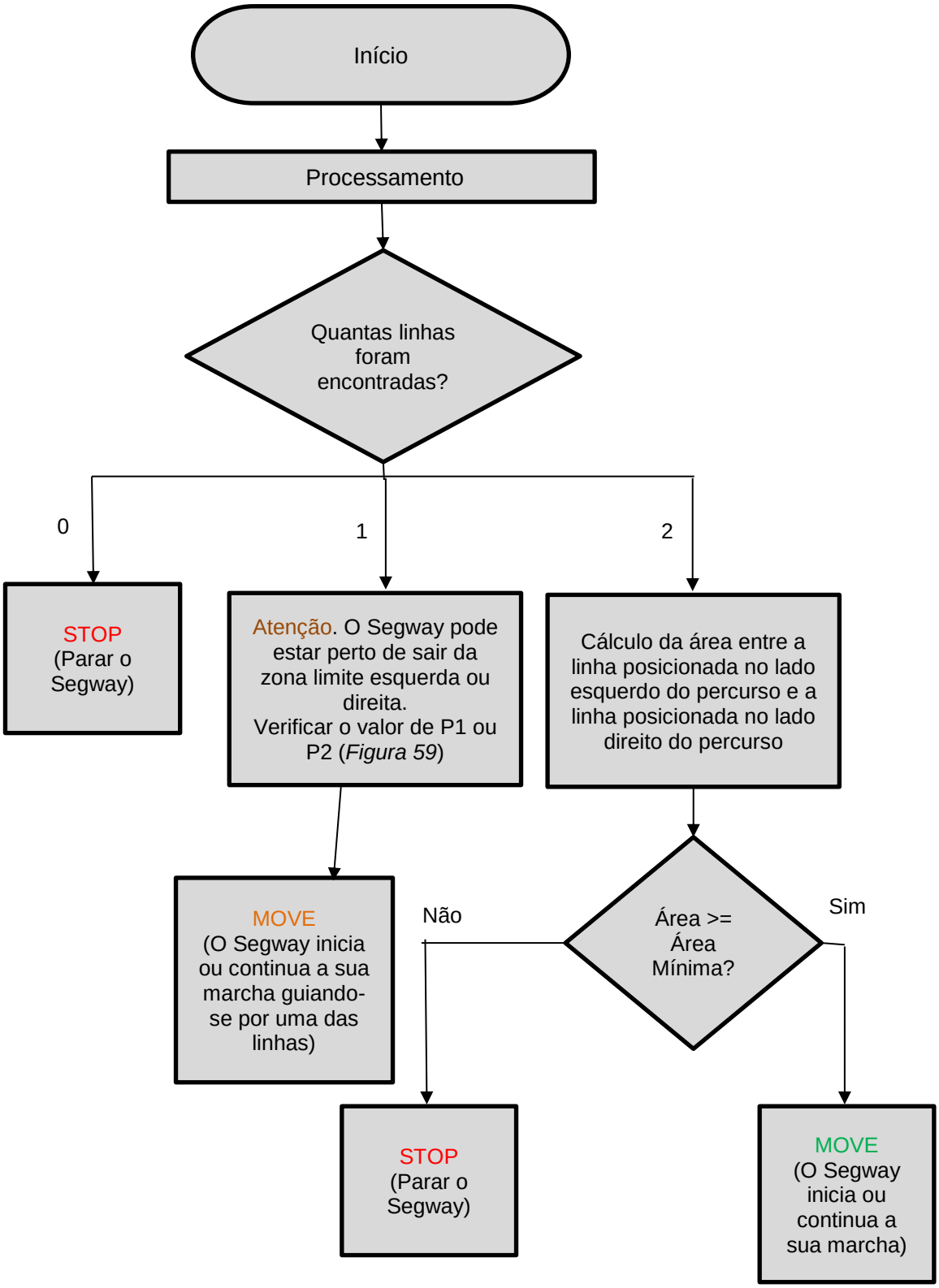


Figura 56 - Fluxograma das decisões a tomar para o novo algoritmo.

4.5. DETEÇÃO DE OBSTÁCULOS

Para deteção de obstáculos é utilizado um sensor distância *RPLidar A2* [24] (*Figura 57*-(a)). De notar que, anteriormente, o Segway não possuia qualquer sensor para deteção de obstáculos, sendo este o principal motivo para muitas das colisões que ocorreram. Estes sensores, são sensores óticos e funcionam atráves do princípio da reflexão de um feixe laiser. Para controlar e fazer a aquisição de dados com o *RPLidar A2* e para controlo de alguns parâmetros, como a frequência do pulso de rotação do motor do *RPLidar A2*, de modo a aumentar o número de pontos medidos pelo rotor scanner, foi utilizada uma biblioteca [25] disponível online.

O "RPLidar A2" está colocado no Segway de modo a que o seu ângulo 0º se posicione ao longo do eixo longitudinal. O sentido positivo do scan laser é no sentido dos ponteiros do relógio (*Figura 57*-(b)). Este equipamento encontra-se caracterizado na *Tabela 9*.

Alcance(m)	0.2 – 18
Alcance(º)	360
Medições por volta	400
Comprimento de onda do feixe laser	775-795
Consumo (mA)	6
Ambiente de operação	Interior (<1000lux)

Tabela 9 - Principais características do RPLidar A2 [24]

a) b)

Figura 57 - (a)RPLidar A2. (b)Sistema de coordenadas. [24]

[24] RPLIDAR A2, "Introduction and Datasheet", 2016
[25] @GitHub, *newpavlov* , "Python module for RPLidar A1 and A2 rangefinder scanners"

Os principais obstáculos que surgem são:

- Operadores
- Veículos
- Elementos pertencentes à estrutura da fábrica (Paredes, Barras, pilares, armários, portas, etc) (*Figura 58*-(c) e (d)).

Os obstáculos mais frequentes são os veículos que frequentemente estão estacionados junto ao caminho do segway ou sobre o mesmo (*Figura 58*-(a),(b) e (e)). Por vezes, também circulam no mesmo percurso, operários (*Figura 58*-(f)), sendo necessário estabelecer regras, para que haja uma livre circulação em segurança. Na Tabela *10* estão escritas as regras. A única situação em que o Segway pode iniciar a marcha corresponde ao momento em que ele deteta as linhas e não tem qualquer obstáculo dentro do intervalo pré-definido.

Deteta obstáculo e vê linha	STOP e espera 3 seg
Deteta obstáculo e não vê linha	STOP
Não deteta obstáculo e vê linha	MOVE
Não deteta obstáculo e não vê linha	STOP

Tabela 10 - Hierarquia para a tomada de decisão.

Figura 58 - Obstáculos frequentes.

Posto isto, foi desenvolvido um algoritmo que permitisse ao Segway a tomada de uma decisão, sempre que um destes obstáculos surgisse no seu caminho (algoritmo disponível no *ANEXO 4)*. Foi estabelecido um intervalo ângular e uma distância mínima (*Equação 19, Equação 20, Figura 59*), para que o Segway se imobilizasse caso algum obstáculo interferisse no seu caminho. No manual do fornecedor do sensor de distância [24], apenas é assegurada a medição com precisão para corpos brancos. Para comprovar isto, foram feitos testes com veículos de diferentes cores e configurações (com ou sem parachoques) (*Figura 60*).

Definiu-se como distância mínima um valor de 1000mm (*Equação 20*), para o Segway ter tempo de se imobilizar sem que atingisse o obstáculo. Como os parachoques frontais dos carros também eram escuros, foi alterado o plano de inclinação do sensor (*Figura 61*), para que o feixe laizer conseguisse atingir uma área, que não pertencesse ao parachoques, caso o veículo se posicionasse no seu caminho. Deste modo conseguiu-se que o RPLidar A2 detetasse o veículo.

Para tornar mais fácil a visualização da distância a que o obstáculo se encontrava, aparece no canto superior esquerdo da imagem o valor da distância ao obstáculo quando este é detectado (Figura *62*).

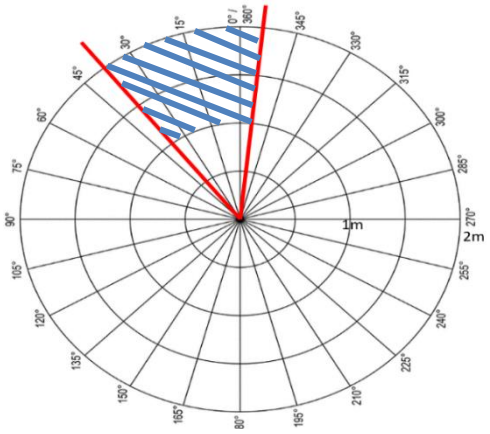


Figura 59 - Representação gráfica do intervalo de deteção dos obstáculos

$$-40 < \ \theta(º) \ < 5 \tag{19}$$

$$1000 < distância(mm) < 2000 \tag{20}$$

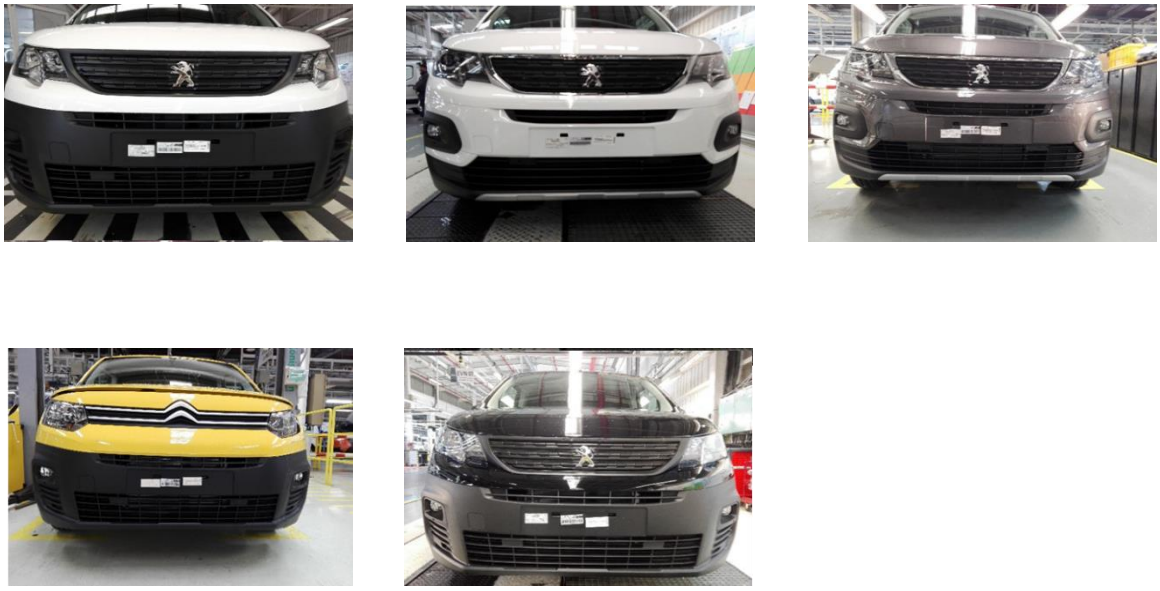


Figura 60 - Diferentes tipos e configurações de veículos.

Figura 61 - Variação do ângulo de inclinação do plano de medição de distâncias do sensor.

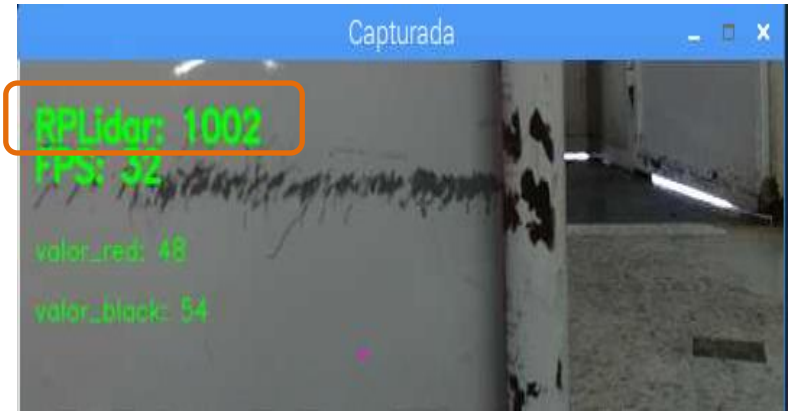


Figura 62 - Distância lida ao obstáculo mais próximo.

4.6. ALGORITMO DE CONTROLO DE DIREÇÃO

Os principais objetivos do controlo da direção são:

- Definir em cada momento qual a direção a tomar pelo Segway.
- Dirigir o Segway numa trajetória estável sem grandes oscilações.

O algoritmo inicial, apresentado na *secção 2.2.3,* tem um algoritmo de controlo de direção associado. Consoante o valor da variável *distância (*distância em pixels desde o ponto central de referência na imagem até ao píxel pertencente aos contornos da linha) se enquadre num certo patamar, os atuadores vão assumir um determinado curso (chama-se a isto um controlo por camadas).

No entanto o controlo que estava a ser utilizado tornava-se, por vezes, instável e, por isso, o Segway saia frequentemente dos seus limites, pondo em risco a sua segurança e a dos operadores. Na Figura 63, temos representados os vários valores de distância registados ao longo do percurso, que segundo o algoritmo da secção 2.2.3, corresponde ao valor em pixels da distância entre o centro da imagem até ao primeiro píxel encontrado. Na mesma figura podemos ver a vermelho o momento em que o Segway entra em regime instável, acabando por sair dos limites com valores próximos dos 300 píxeis para a variável *distância*.

O facto do Segway se tornar instável poderá ter por causa os problemas que abordamos ao longo desde capítulo (iluminação e estado do piso), mas também pode ter por causa um mau controlo de direção desenvolvido pelo programador. Por esta razão foi melhorado o algoritmo de controlo do Segway e os resultados serão mostrados no próximo capítulo, capítulo cinco. No *ANEXO 5* encontra-se o novo algoritmo elaborado para melhorar a trajetória direcional do Segway.

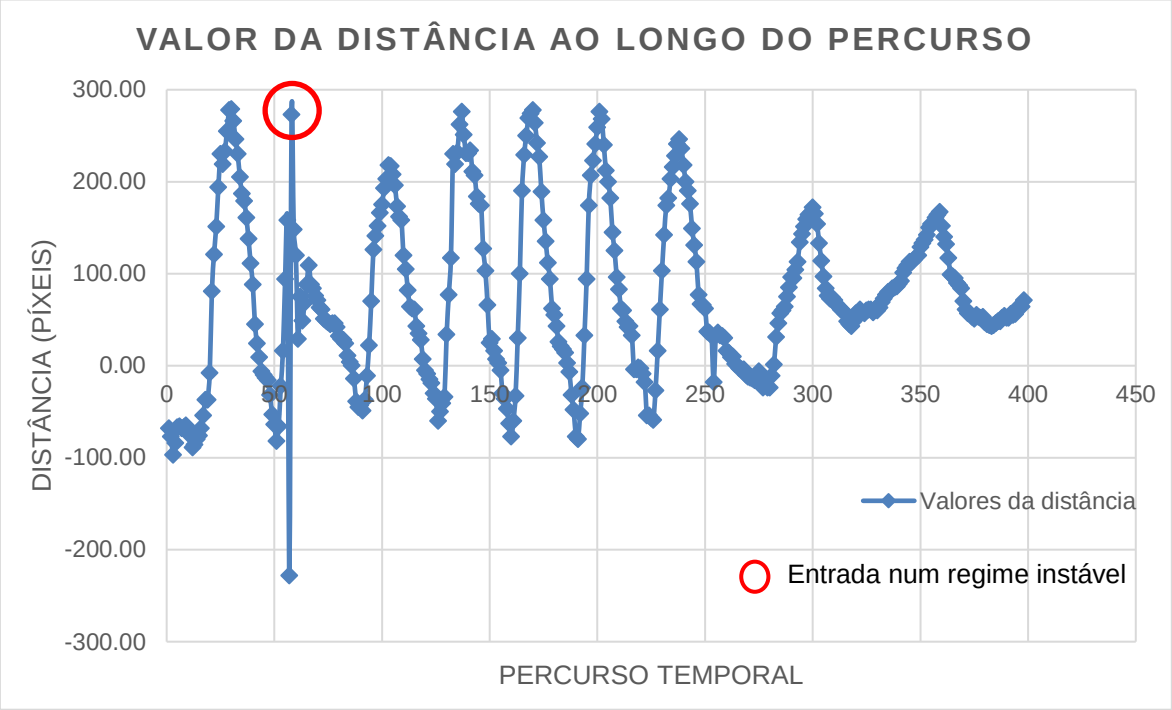


Figura 63 - Valores da distância aos contornos da linha, ao longo do percurso.

4.7. CARACTERIZAÇÃO DOS CONSUMOS DE ENERGIA

Atualmente todos os equipamentos que são utilizados para guiar autonomamente o segway são alimentados por apenas uma das baterias do segway. Isto leva a um consumo desnívelado entre as duas baterias, estando sempre uma com menos carga do que a outra.

É importante que o conjunto das duas baterias que constituiem o Segway, tenham autonomia suficiente para fazer um turno completo de oito horas. Cada bateria [26] consegue debitar cerca de 5200 mAh por hora e o consumo médio para cada equipamento está registado na *Tabela 11*. No total, todo o sistema é fornecido por 10400 mAh, sendo este equivalente a 8 níveis de bateria, que surgem no mostrador da Figura *64*.

Nos consumos, o "RPLidar" não foi considerado, pois está ligado pela porta USB ao RaspberryPi3 B+, como será com qualquer equipamento que esteja ligado às baterias, indiretamente, por via de outro equipamento.

Para quantificar o débito de energia por unidade de tempo e por unidade de produção são registados, ensaio a ensaio, o número de traços (um traço corresponde a um nível), que o monitor mostra de hora a hora. Os resultados obtidos para o consumo durante um turno completo estão apresentados e analisados no capítulo seguinte (capítulo cinco).

[26] Segway©," Lithium-ion (Li-ion) Batteries: Important Information", 2013

Principais equipamentos de consumo	Consumo(A)
RaspBerry	0.3
Motorores Dc(2x) Horizontal	6
Motores Dc(1x) Vertical	3
Segway	--

Tabela 11 - Consumo médio dos equipamentos que se ligam às baterias [8],[9].

Figura 64 - Monitor onde é mostrado o nível de bateria.

4.8. MANUTENÇÃO E CUIDADOS A TER COM O SISTEMA

Por último e não menos importante, a segurança. Como já foi referido no capítulo número dois, aconteceram alguns acidentes enquanto os operadores controlavam o Segway, nomeadamente quedas. Estes acontecimentos poderiam ter sido evitados se algumas medidas fossem previamente tomadas. Por isso foi desenvolvido um conjunto de regras e hábitos para aplicar durante o procedimento das operações com o Segway e que estão tabelados na Tabela *12*. Com a aplicação destas regras pretende-se diminuir o número de acidentes com os operadores ou a gravidade do acidente.

Regras Diárias
Sempre que iniciado o turno, é necessário garantir pelo menos 7 a 8 traços de bateria no mostrador (*Figura 64*).
É necessário a existência de um par de baterias em stock, completamente carregados, prontos a serem inseridos no Segway na troca de turnos.
O Segway não pode iniciar o seu ciclo normal no turno seguinte sem antes mudar o par de baterias.
Verificar o estado da pilha que alimenta o mostrador do segway antes de iniciar o turno.
Qualquer operador que utilize o segway durante a sua locomoção deve usar os EPI's apropriados para o efeito.
Se o piso estiver molhado ou em mau estado o operador deve circular com velocidade moderada ou reduzida (<4km/h)

Se for necessário imobilizar o segway por algum motivo é importante desligar todas as fontes de alimentação do sistema.
Antes de iniciar o turno é necessário verificar as condições em que se encontra o piso e a linha marcadora do solo.
Caso, por algum motivo, o número de traços no mostrador seja igual ou inferior a 2 níveis, é necessário que o Segway seja imobilizado e efectuada a troca de baterias (*Figura 64*)

Tabela 12 - Tabela dos cuidados a ter com o Segway diariamente.

A nível da manutenção que é necessário ter com o equipamento, foi construida uma tabela de manutenção, disponível no *ANEXO 8*. Nesta tabela são definidas as referências das peças a manter, o período de manutenção e a atuação períodica que se deve realizar. A tabela de manutenção é importante para que o sistema funcione em condições normais, evitanto assim paragens por longos períodos de tempo, caso algum material atinja o seu limite de fadiga/rutura.

A principal vantagem que se ganha com a elaboração destas medidas preventivas tem efeito no tempo médio entre falhas (*TMBF*). Isto é, se o número de acidentes diminuir e o tempo de paragem do Segway, por mau estado de um ou outro equipamento, também diminuir, então o *TMBF* irá aumentar, sendo este um ponto positivo.

5. RESULTADOS

No presente capítulo são apresentados os resultados obtidos nos ensaios e os impactos das alterações que se fizeram e que foram propostas no capítulo anterior (capítulo quatro). Esta etapa é importante, pois, é com base numa análise quantitativa e qualitativa das pequenas alterações realizadas, que se tiram as conclusões gerais e representativas do desempenho atual do Segway, ao fim destes três meses e meio de estágio.

5.1. ILUMINAÇÃO AO LONGO DOS SETORES DA BTU

Na *sub-secção 4.1* foi desenvolvido um equipamento que media as variações de luminosidade. Este equipamento foi colocado ao longo das três secções (Secção A, B e C) nas quais se dividiu o percurso a efectuar pelo Segway.

Os ensaios para obter os dados da Figura 65, foram realizados no mês de Abril de 2019, altura em que o Sol se põe, por volta das 19 horas. O objetivo deste ensaio é mostrar que a luminosidade que se faz sentir no chão de fábrica, quando o Segway está em operação, não é constante e sofre inúmeras variações ao longo dos três turnos. É também importante relevar que os ensaios foram feitos em iguais condições meteorológicas exteriores ao longo de três dias (um dia para cada setor).

Analisando o gráfico da Figura 65, podemos concluir que a intensidade luminosa, que se faz sentir ao longo dos três setores varia e nunca é igual nos três. O setor B, é o setor com valor de luminosidade médio maior. No entanto, o setor C, é o que sofre a maior variação de amplitude nos valores de iluminação.

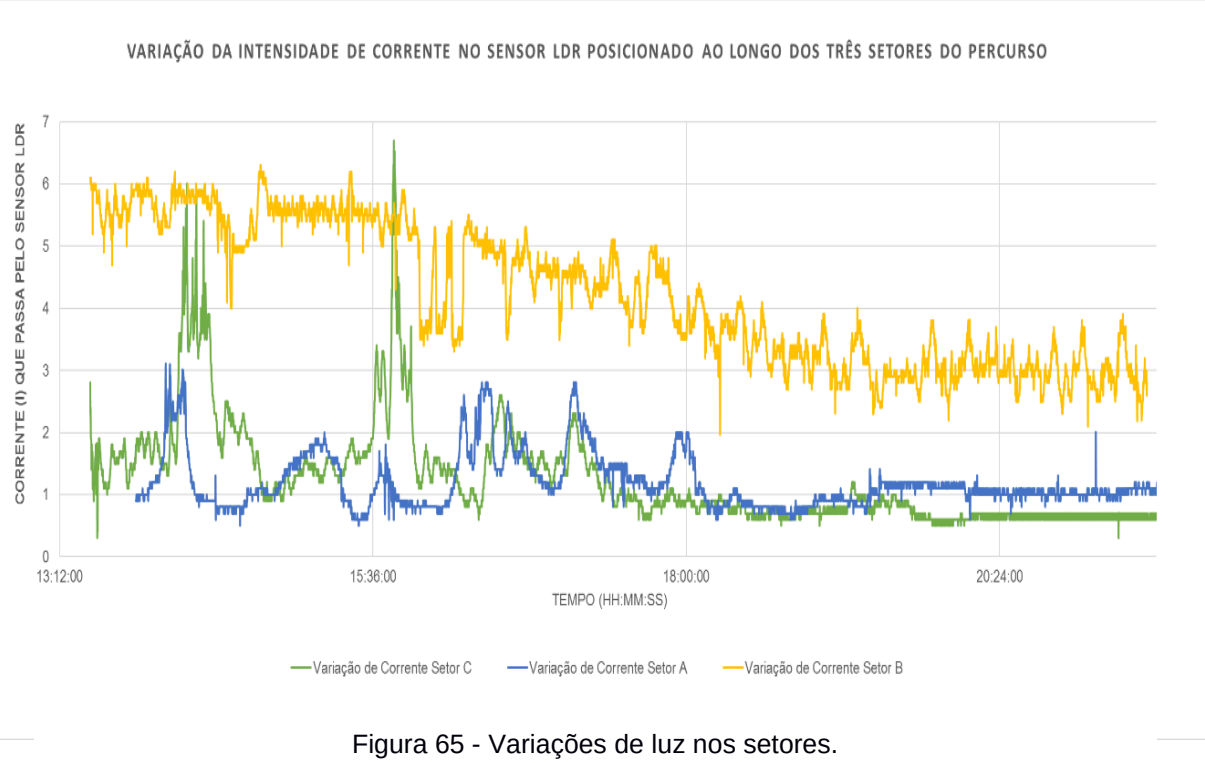


Figura 65 - Variações de luz nos setores.

Durante o período da noite (a partir das 19 horas), os valores medidos nos três turnos deixaram de sofrer grandes variações de amplitude, oscilando de uma maneira quase períodica em torno de um dado valor de intensidade. No mesmo período, o setor C, é o que tem o valor mais baixo de intensidade luminosa, facto este que se justifica pelo baixo número de lâmpadas que existem neste setor.

	Período do dia		
	Setor A	Setor B	Setor C
Falhas	10	1	2
	Período da noite		
	Setor A	Setor B	Setor C
Falhas	2	10	0

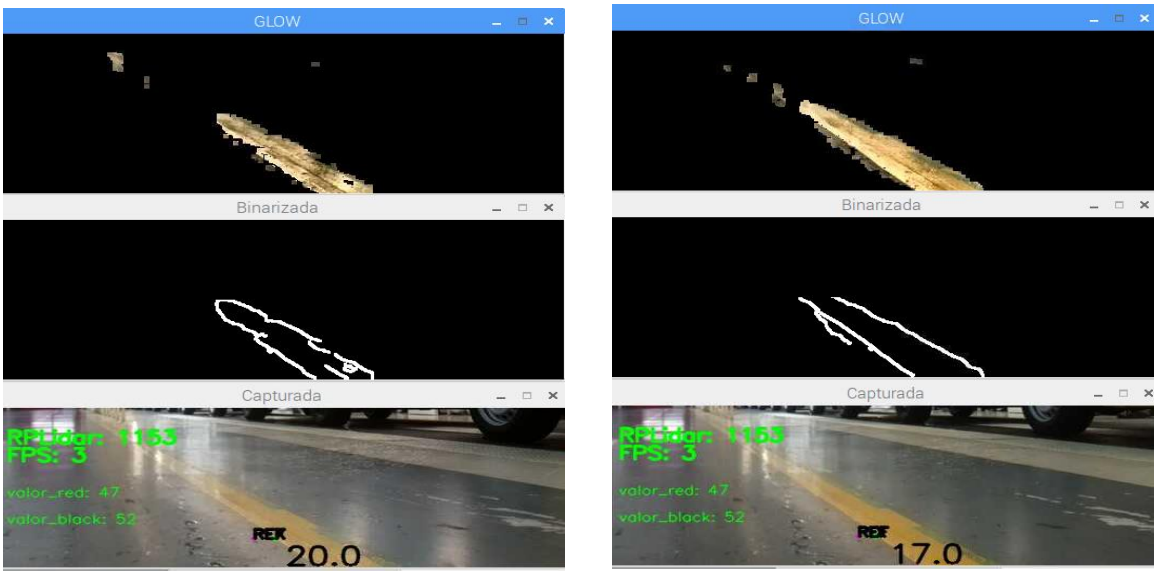
Tabela 13 - Número médio de falhas ao longo dos setores.

Depois de realizados alguns ensaios constata-se que o setor onde se registam mais falhas, durante o período do dia, é o setor A (Tabela *13*), facto que pode ser justificado pela maior número de variações de luminosidade ao longo das horas (*Figura* 65) .

Durante a noite, o setor com menos falhas é o setor C (*Tabela 13*) e, analisando mais uma vez o gráfico da Figura 65, podemos concluir que este menor número de falhas se deve à ausência de variação de luminosidade, contrariamente ao setor B, que é onde acontece o maior número de falhas durante o período da noite, por causa das constantes variações de luminosidade.

5.2. IMPACTO DA APLICAÇÃO DE UMA LENTE COM FILTRO POLARIZADOR

O uso de um filtro polarizador na lente da PiCamera V2 tem como principal objetivo diminuir a quantidade de luz que atinge o sensor da câmara, aumentanto os contrastes na imagem que se obtém. Após alguns ensaios experimentais e, pela análise da Figura *66*, podemos concluir que com o uso de um filtro polarizador, na imagem binarizada da Figura *66*-(b), os contornos da linha estão mais definidos, comparativamente com os contornos da linha binarizada sem o uso de qualquer filtro Figura *66*-(a).



a) b)

Figura 66 - Impacto do filtro polarizador.

Os contornos da linha estão mais definidos quando se usa um filtro polarizador uma vez que, os contrastes entre a linha e o piso aumentaram. Está assim provado que o uso de um filtro polarizador é essencial para melhorar a segmentação da linha.

5.3. IMPACTO DA COLOCAÇÃO DE UMA LINHA AMARELA NOVA NO LADO DIREITO DO PERCURSO

No *sub-secção 4.2*, foi abordado o mau estado em que se encontrava a linha e os efeitos que este causava no algoritmo apresentado na *sub-secção 2.2.3*. Pelas razões que foram apresentadas, foi colocada no piso uma nova linha de cor amarela sobre as linhas amarelas e verde que anteriormente existiam. Neste capítulo é demonstrado o impacto que esta alteração teve no mesmo algoritmo de segmentação da linha.

Analisando as imagens da Figura 68, concluimos que a segmentação dos contornos da linha teve melhorias em relação ao estado anterior (*Figura* 30, *sub-secção 4.2*).

O facto dos contornos das linhas (imagens binarizadas da *Figura 68)* estarem melhor definidos, tem uma influência positiva para o novo algoritmo desenvolvido, porque caso não se assemelhem a uma reta, o algoritmo não vai conseguir detetar as linhas na imagem.

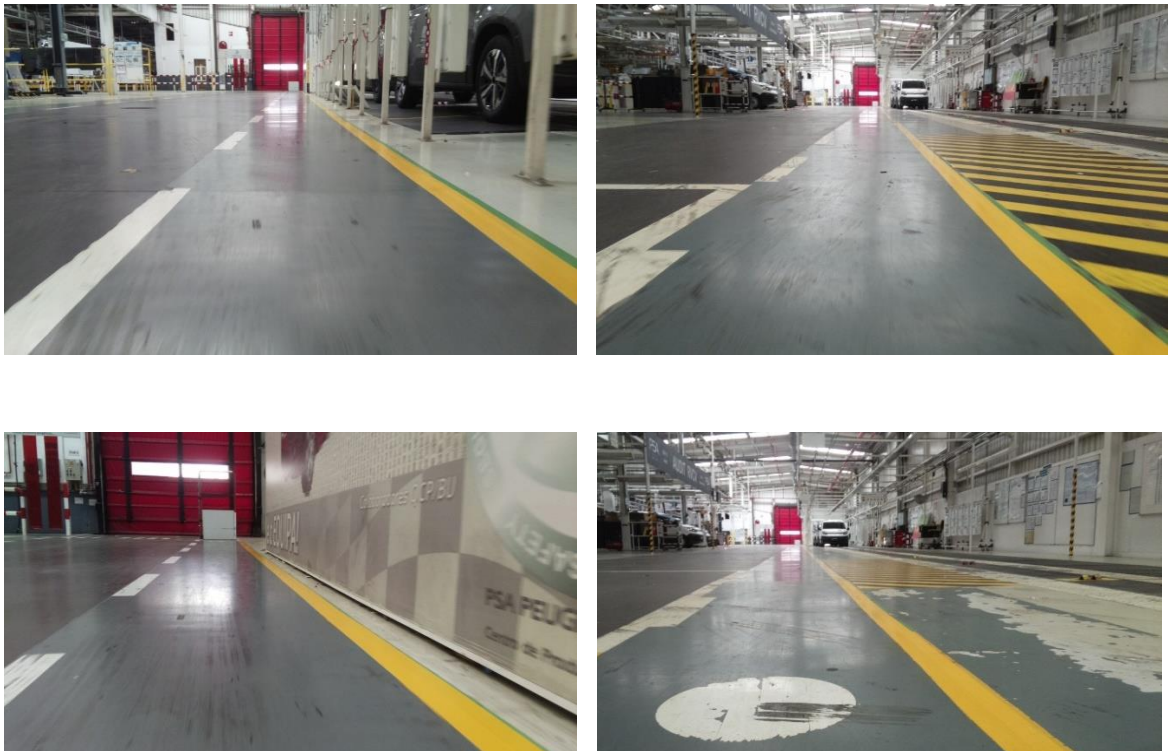


Figura 67 - Imagens do percurso com a nova linha amarela.

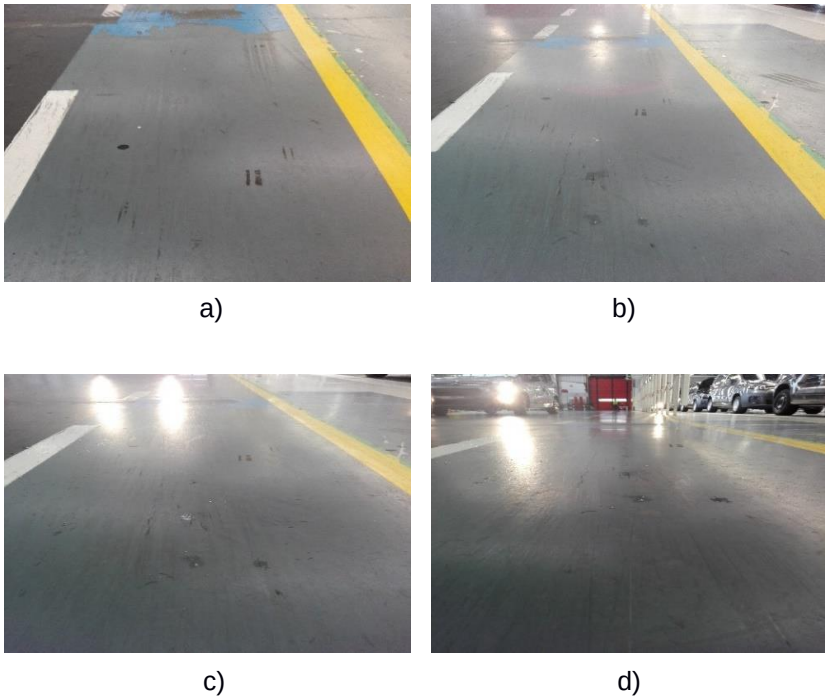
Figura 68 - Impacto na segmentação da linha amarela após as melhorias feitas no piso

Comparando as imagens binarizadas da Figura 68 com as imagens binarizadas da Figura *30* (*sub-secção 4.2),* facilmente se percebe que na Figura *68* os contornos da linha estão melhor definidos. Com o algoritmo acontece o mesmo. Por isso é que se consegue atingir ótimos valores na precisão com que se faz a segmentação da linha, quando esta se encontra em bom estado (limpa e sem marcas), como se ilustra na Figura *68*.

5.4. IMPACTO DA POSIÇÃO E ORIENTAÇÃO DA CÂMARA NO CONTRASTE DA LINHA

Com a colocação da linha amarela nova no piso, a quantidade de reflexos que atingia a câmara aumentaram. Este facto surgiu porque a linha estava limpa e sem marcas, o que aumentou a sua refletância (quantidade de luz que a linha reflete). Consequentemente, aumenta o número de vezes que o Segway não consegue detetar a linha. Para reduzir o impacto que os reflexos estavam a ter na segmentação da linha, testou-se a captura de imagem em diferentes posições da câmara (alteração da altura e inclinação).

Independentemente do tipo de câmara usado, podemos concluir que com o aumento da inclinação da câmara, o contraste da linha aumenta, porque diminuimos a exposição da camera à luz (ou seja, quantidade de reflexos que inside no sensor da câmara é menor). Na Figura *69,* podemos reparar que quando a inclinação é maior, Figura *69*-(a), a quantidade de reflexos que inside no piso e se reflete na câmara é menor. No sentido oposto, temos a imagem da Figura *69*-(c)*,* onde a inclinação é menor e o contraste entre a linha e o piso diminui devido à interferência dos reflexos provenientes do meio exterior à câmara.



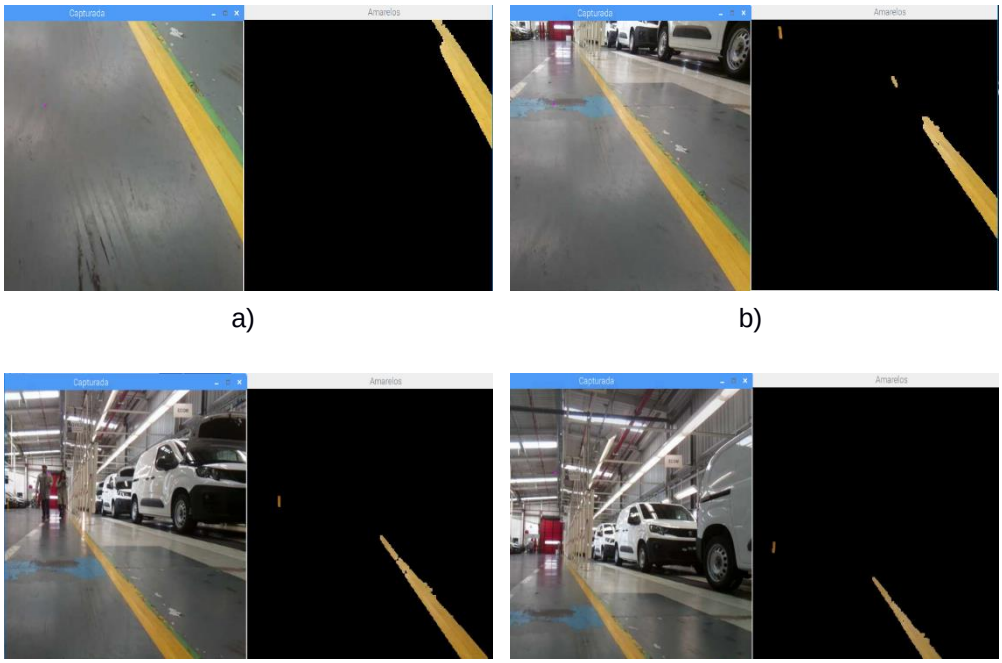
a) b)

c) d)

Figura 69 - Variação da posição da câmara.

Na Figura *70*, podemos ver o impacto no contraste da linha e posterior segmentação da sua cor para diferentes inclinações da câmara. Conclui-se que uma grande variação de inclinação num sentido ou noutro não trás os melhores benefícios, pois, por um lado, podemos aumentar o número

de reflexos e por outro podemos deixar de ver a linha a segmentar, porque a câmara está muito focada para o chão (*Figura* 70-(d)*)*. Por isso, é importante encontrar um ponto médio de equilíbrio para a posição da câmara.



a) b)

c) d)

Figura 70 - Diferentes níveis de inclinação da câmara. (a) +2; (b) +1; (c) 0; (d) -1;

Para além de alterar o impacto que a iluminação tem durante a captura de imagem, a mudança de inclinação e posição da câmara tem impacto nos valores que são usados para controlar a direção do algoritmo. Analisando a Figura *71* (imagens tiradas no mesmo local, no mesmo instante de tempo), podemos verificar que para ângulo de inclinação (-2; 0; +1 ou +2) o valor da varíavel *distancia* varia (-132; -71; 131; 24). É importante por isso, definir qual a melhor posição da câmara para evitar que se esteja constantemente a fazer alterações no algoritmo de controlo de direção.

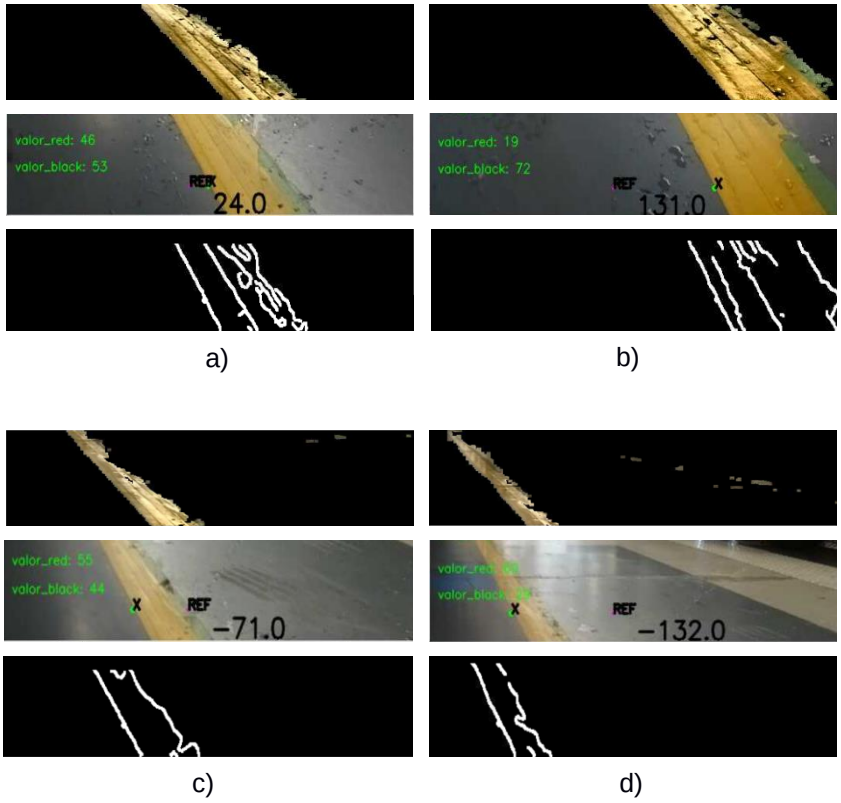


Figura 71 - Valores de distância para diferentes inclinações da câmara (a) +1 (b) +2 (c) 0 (d) -2

5.5. VANTAGENS DO NOVO ALGORITMO DESENVOLVIDO

Depois de desenvolvido um novo algoritmo na sub-secção 4.3.2. é importante que este consiga ultrapassar uma série de circuntâncias em que o algoritmo antigo (*sub-secção 2.2.3*) apresentava falhas. Neste sub-capítulo, são analisados alguns destes momentos e assim se justifica o facto de que a implementação do novo algoritmo é importante para ultrapassar os principais problemas sentidos.

Comparando os dois algoritmos, enumeramos em seguida as vantagens que o novo algoritmo apresenta em relação ao algoritmo antigo:

1. Apesar do algoritmo segmentar a cor dos coletes laranja, estes não interferem na segmentação e representação das linhas (*Figura 72) ;*
2. A cor dos coletes esverdeados não é detetada e não interfere na construção da linha virtual (*Figura 74) ;*
3. As cores das roupas dos operadores não são segmentadas juntamente com a cor da linha (*Figura 76)* ;
4. A linha é segmentada mesmo estando alguém a obstruí-la (*Figura 75) ;*

5. Precisão com que são segmentadas as cores e contornos da linha e a construção da linha virtual sobre a imagem final (*Figura 73) ;*
6. Precisão na representação da linha, mesmo com carros amarelos presentes no lado esquerdo do percurso (*Figura 78)*. Na *Figura 79*, podemos ver, na imagem binarizada, que os reflexos são detetados pelo filtro *Canny*, no entanto não tem a geometria suficiente para ser considerada um segmento de reta e a linha virtual não é desenhada *;*
7. Independentemente das portas estarem abertas ou fechadas a segmentação da linha é feita sem problemas (*Figura 77) ;*
8. Os efeitos da iluminação durante o período noturno não afetam a precisão com que são detetadas as linhas, após se ter alterado a posição da câmara (*Figura 80 e Figura 81) ;*

Figura 72 - Coletes Laranjas

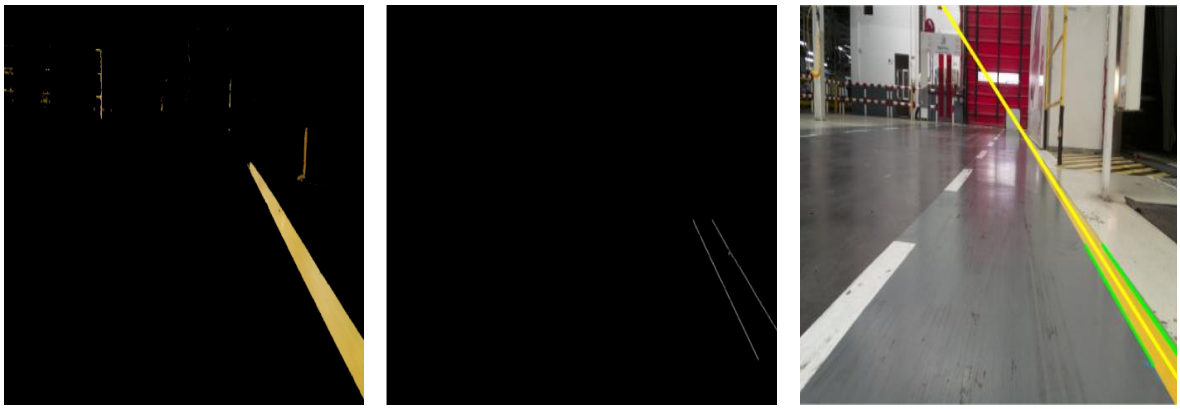
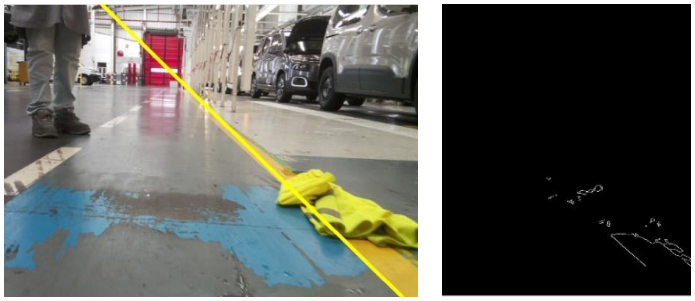
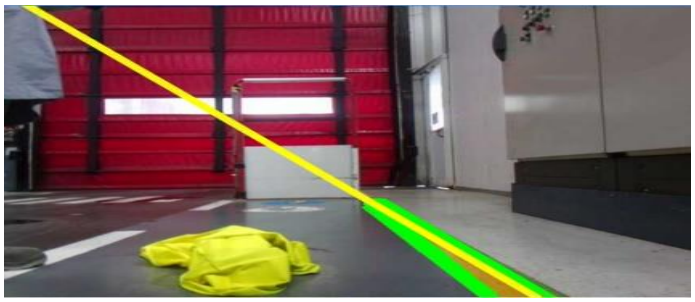


Figura 73 - Precisão na representação da linha virtual.

Figura 75 - Obstrução da linha.



a)



b)

Figura 76 - Coletes amarelos

Figura 74 - Roupa dos operadores.

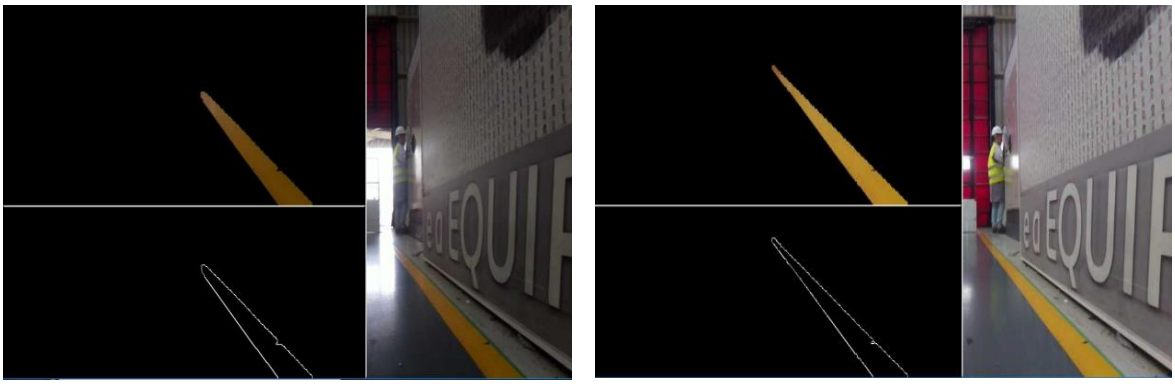


Figura 77 - Segmentação da linha com a porta aberta e a porta fechada.

Figura 79 - Representação da linha virtual sobe a presença de algumas condicionantes.

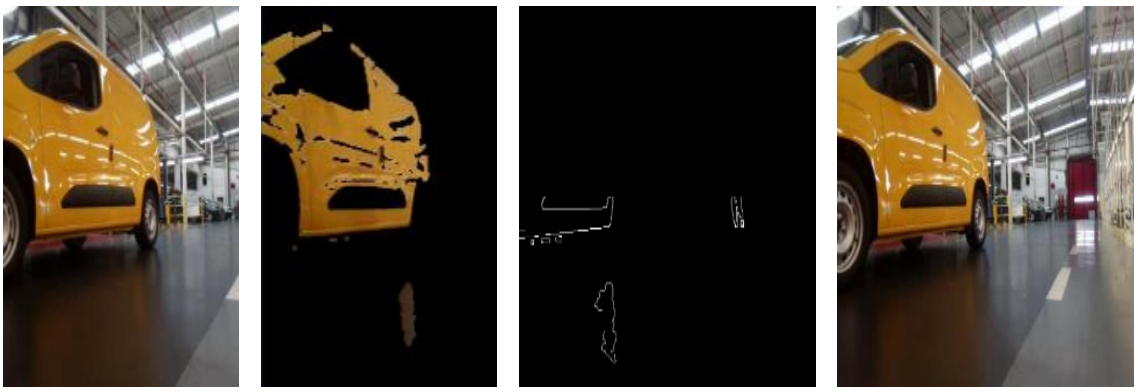


Figura 78 - Reflexo de carros amarelos

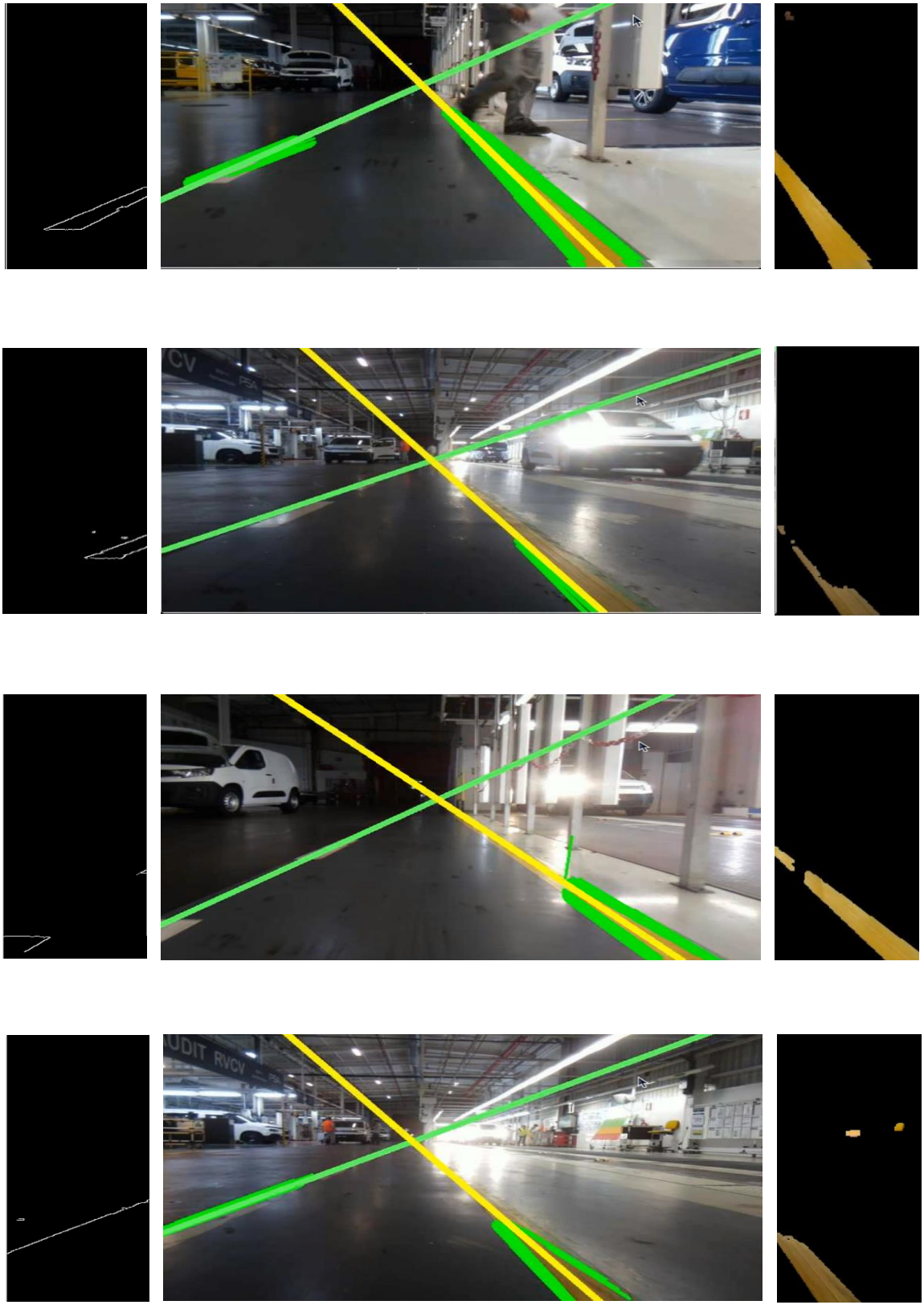
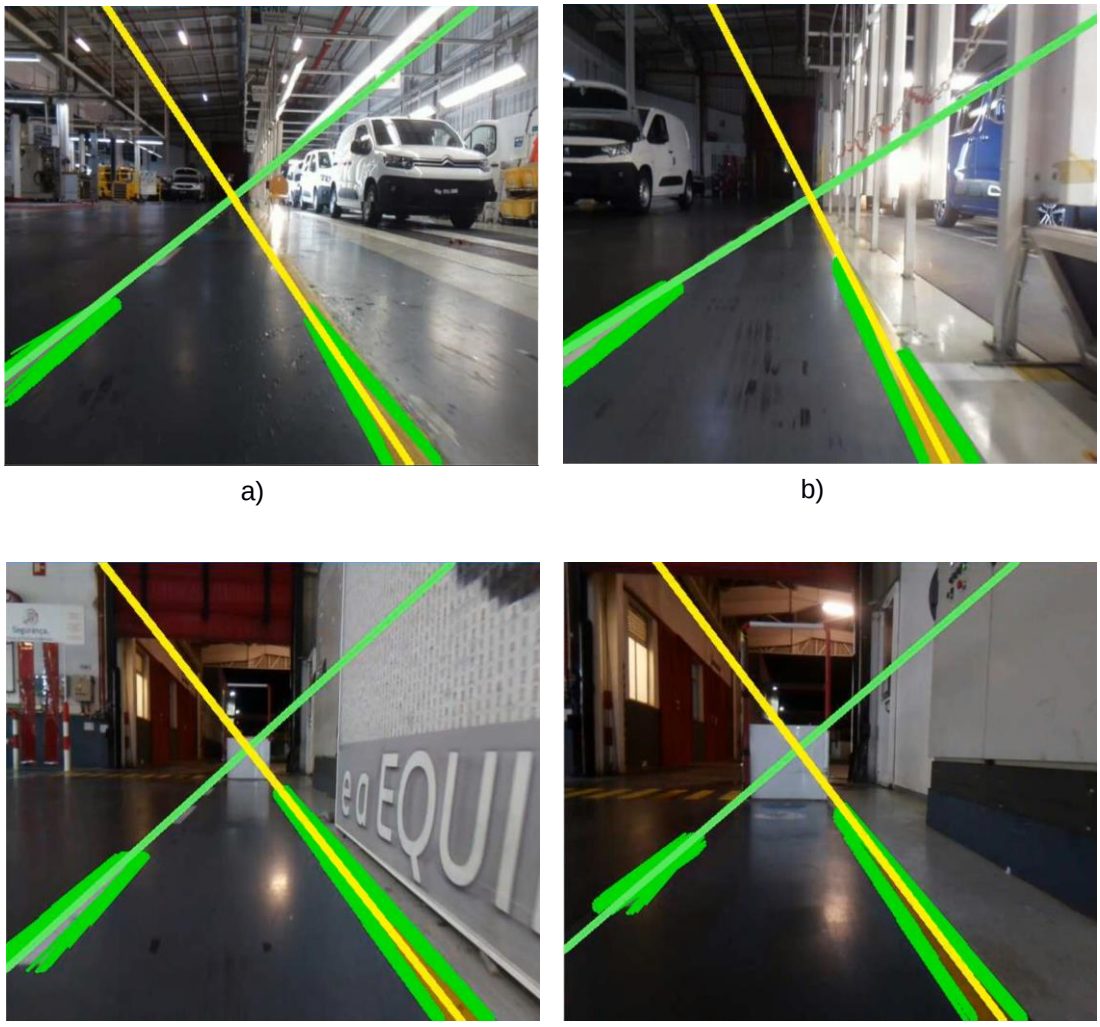


Figura 80 - Iluminação durante o turno da noite.



a) b)

c) d)

Figura 81 - Iluminação durante o turno da noite.

5.6. IMPACTOS NO TEMPO ÚTIL E VELOCIDADE DO SEGWAY COM ALTERAÇÃO DO CURSO MÍNIMO DO ATUADOR VERTICAL

A velocidade com que o Segway realiza o percurso autónomo tem efeitos no tempo que demora a chegar ao final da linha. Esta velocidade é controlada pelo atuador vertical.

Quanto mais inclinado (menor o curso do atuador), maior a velocidade atingida. No entanto, como anteriormente foi dito (*sub-secção 2.2.2*), a máxima inclinação que o segway obtêm, sem que perca o equilíbrio, corresponde a uma inclinação de 7º da haste, aproximadamente em relação ao ponto neutro onde o Segway se encontra imobilizado. Para perceber se o Segway poderia aumentar a sua velocidade ao longo do percurso autónomo, podendo poupar tempo útil durante o trajeto, foi feito um teste de velocidade, onde se obteve os resultados mostrados na Tabela *14*.

Distância(m)	Curso Vertical	Tempos(s)	Tempo médio (s)	Velocidade (m/s)	Velocidade (km/h)
5,67	65	12,2	12,08	0,47	1,69
	65	12,11			
	65	12,07			
	65	11,92			
	61	9,52	9,76	0,58	2,09
	61	9,67			
	61	10,26			
	61	9,6			
	60	8,96	9,09	0,62	2,25
	60	9,13			
	60	9,13			
	60	9,14			

Tabela 14 - Teste de velocidade.

Para uma distância constante de 5.67 metros foram registados os tempos médios que o Segway demora a percorrer nessa mesma distância. Ao longo dos ensaios, variou-se o curso do atuador vertical, o que provocou alterações no tempo final do Segway. Estas alterações do curso, como era de esperar, têm influência direta na velocidade final de deslocamento. Na Tabela *14*, estão quantificadas as velocidades.

A verde, é representada a velocidade atual do Segway e a Vermelho, é a velocidade máxima a que consegue andar sem que o controlador central o imobilize. Não é possível atingir valores maiores de velocidade pois o controlador de equilíbrio, que está embutido no sistema do Segway, desliga-se automaticamente quando uma certa inclinação (10 ± 5º) é atingida.

Após realizados alguns cálculos, foi possível poupar cerca de 28 segundos por volta, se o Segway se movimentar a 2.09 km/h, e 37 segundos por volta, se o mesmo se movimentar à velocidade máxima. A esta velocidade, no final de um turno, consegue ganhar mais de 1 hora em tempo útil, o que se pode traduzir num aumento de produtividade ao final do mês.

5.7. TESTE DE DETEÇÃO DE OBSTÁCULOS

Para testar a viabilidade do algoritmo de deteção de obstáculos e testar a robustez do sensor ótico (*RPLidar A2*), realizou-se uma sequência de ensaios com veículos de dois tipos, em posição frontal ao Segway. Um véiculo de cor clara (branco) e outro de cor escura (preto). Foi utilizada uma biblioteca [25], disponível online, para marcar os pontos no sistema de coordenadas polares. Os resultados estão plasmados na Figura *82*.

Na Figura *82*-(a), temos os pontos representantes de um carro branco posicionado na zona frontal (0º); na Figura *82*-(b), temos os pontos representantes de um carro preto na zona frontal (0º). Caso não seja detetado qualquer obstáculo, numa distância minima de 15 cm em frente ao sensor, os valores medidos serão considerados iguais à distância minima a medir, ou seja, 15cm.

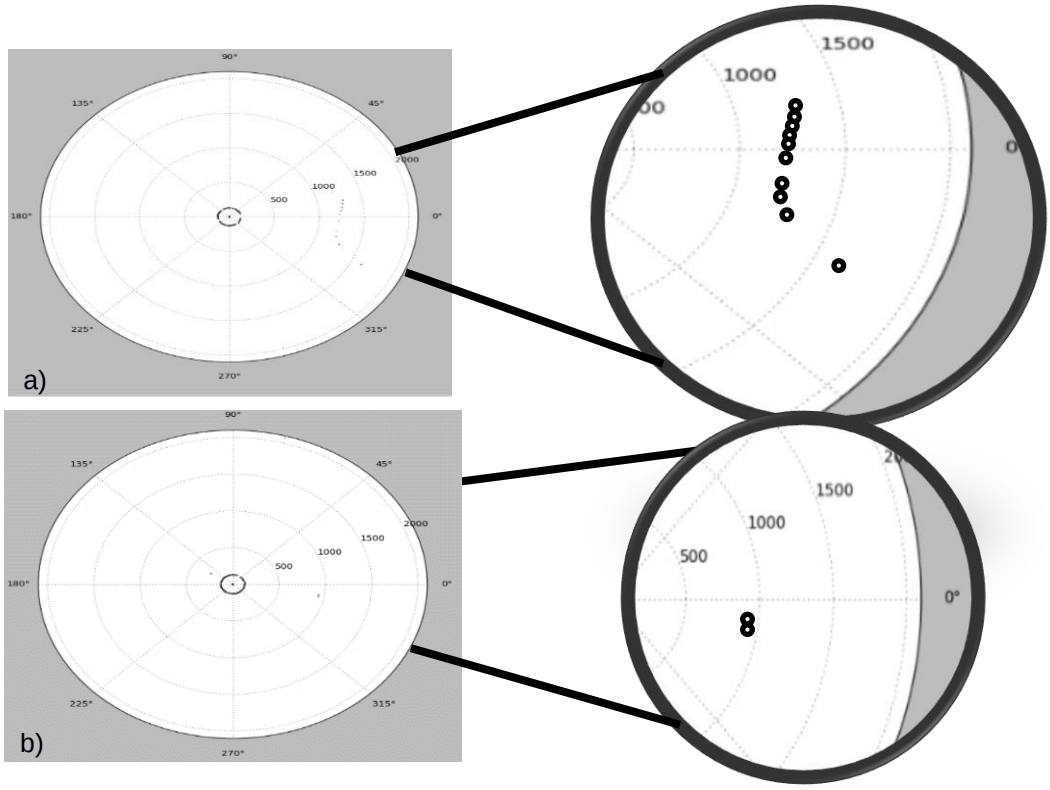


Figura 82 - Medições com o RPLidar A2. a) Carro branco; b) Carro Preto;

Como podemos observar nos gráficos da Figura *82*, na Figura *82-(b)*, o número de pontos é muito reduzido, comparado com o número de pontos encontrados na Figura *89*-A. Este é um dos motivos que leva a que o Segway não tenha detetado os veículos de cor preta, provando mais uma vez que este tipo de equipamento ótico é sensível ao tipo de cor do veículo (factor abordado na *sub-secção 3.2.4*). O mesmo não acontece com as restantes cores menos escuras (brancos e cinzas). Estes ensaios foram feitos com a frequência de rotação do motor do *RPLidar A2* no seu limite máximo, de modo a se detetar o maior número de pontos por rotação (360º).

Um factor que também impossibilita a operação normal do Segway é a leitura de falsos valores (distâncias) num certo intervalo angular, quando na realidade não se encontra qualquer obstáculo dentro do intervalo angular e à distância registada. Como consequência disto, o Segway acaba por se imobilizar inúmeras vezes ao longo do percurso, mesmo sem ter qualquer obstáculo pela frente. Ainda não foi encontrada uma justificação para estas ocorrências, no entanto algumas hipóteses podem ser ponderadas. Nomeadamente o facto deste tipo de sensores óticos ser

fabricado para operar em ambientes interiores com limitações ao nível da densidade de intensidade luminosa que se faz sentir no local. Sendo este um local com fortes varições de luminosidade como já foi provado a *secção 5.1,* o sensor pode estar a sofrer interferência da iluminação ambiente (abordada na *secção 4.1*), levando-o a registar falsas medições.

Concluímos assim que o detetor de obstáculos não é suficientemente robusto, pelos motivos aqui apresentados, sendo necessário a adoção de outra tecnologia para deteção de obstáculos.

5.8. RESULTADOS DOS ALGORITMOS DE CONTROLO DE DIREÇÃO

Na *sub-secção 3.2.3* são apresentados alguns tipos de controlos, no entanto, optou-se por tentar melhorar o controlo por camadas que fora, anteriormente, implantado no Segway. Como o Segway saía dos limites estabelecidos mais que uma vez com frequência, foi estabilizado o seu controlo, ajustando os valores nas camadas de atuação no algoritmo de direção.

Comparando o gráfico da Figura *83* com os dados obtidos na *secção 4.6,* concluímos que com as alterações efetuadas (ajuste dos valores nas camadas) o número de oscilações e a amplitude destas oscilações diminuiu. Antes, as amplitudes dos valores chegavam a ter mais de 200 píxeis de amplitude, e neste momento mal ultrapassam os 100 píxeis. Antes, o Segway saia frequentemente dos limites do percurso acabando por perder a linha e, neste momento, após as alterações, consegue movimentar-se com muito menos oscilações, mais centrado com o centro da pista e saindo menos vezes dos limites.

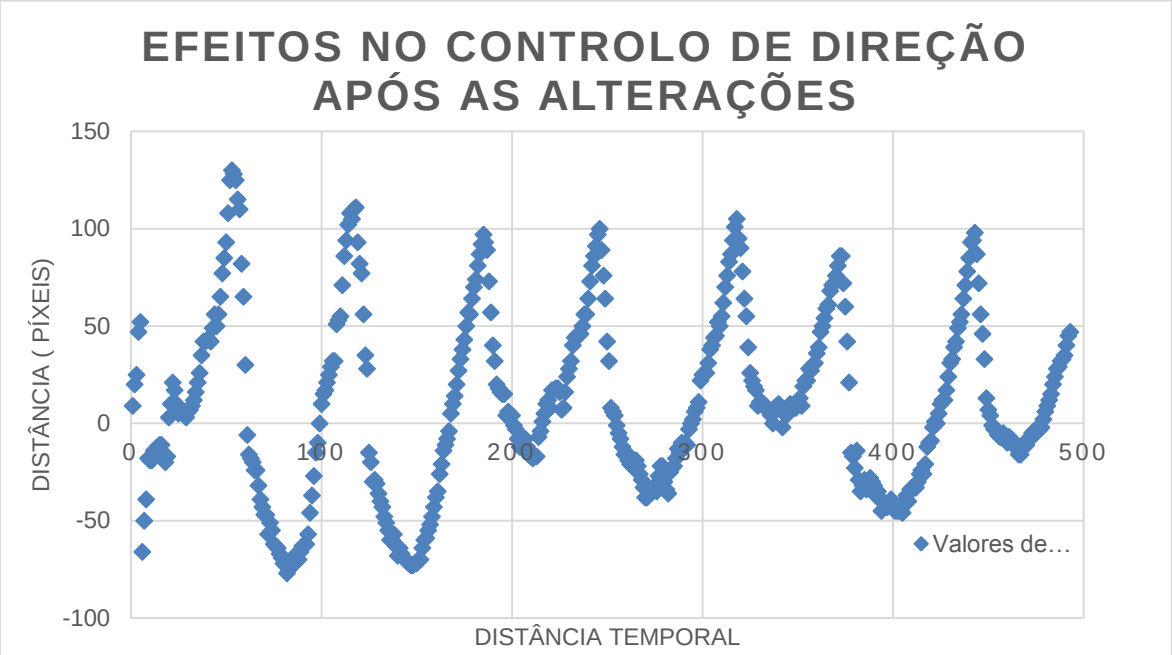


Figura 83 - Valores da distância ao longo do percurso.

Estas melhorias ainda não são significativas para que o desempenho do algoritmo antigo ultrapasse todos os problemas. Como se mostrou ao longo do capítulo cinco, o algoritmo novo aparenta ser muito mais robusto, comparado com o algoritmo antigo e, por isso, é importante que se desenvolva um controlo de direção para este algoritmo desenvolvido.

Na Figura 84, representa-se a distância P1 (distância do centro da imagem ao ponto de interseção da linha amarela com o eixo dos x) e a distância P2 (distância do centro da imagem ao ponto de interseção da linha branca com o eixo dos x). Estes valores de P1 e P2 variam ao longo do percurso.

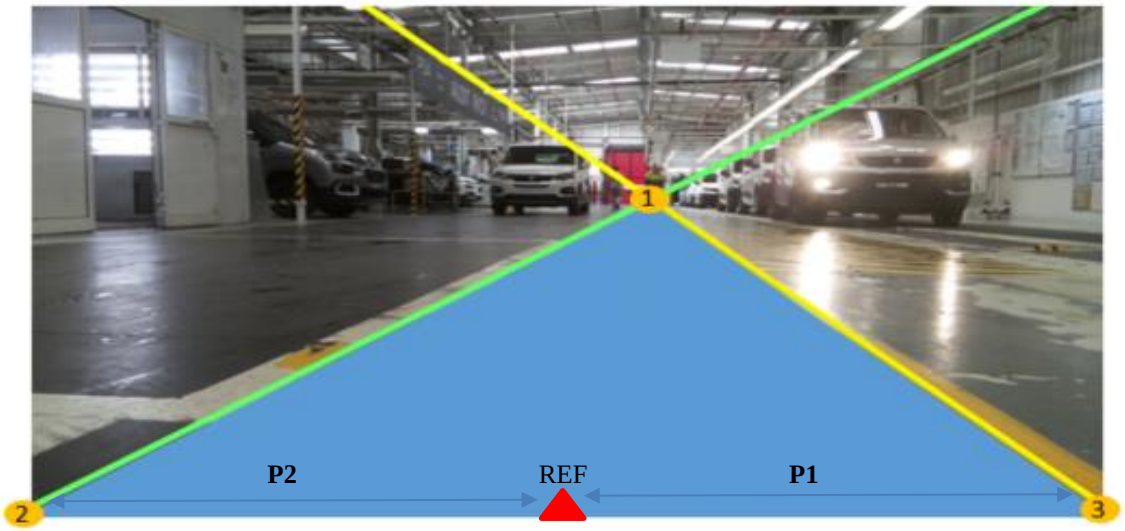


Figura 84 - Valores P1 e P2 para controlo de direção.

Foi feito um registo dos valores de P1 e P2 ao longo de um percurso completo (Figura *85).*

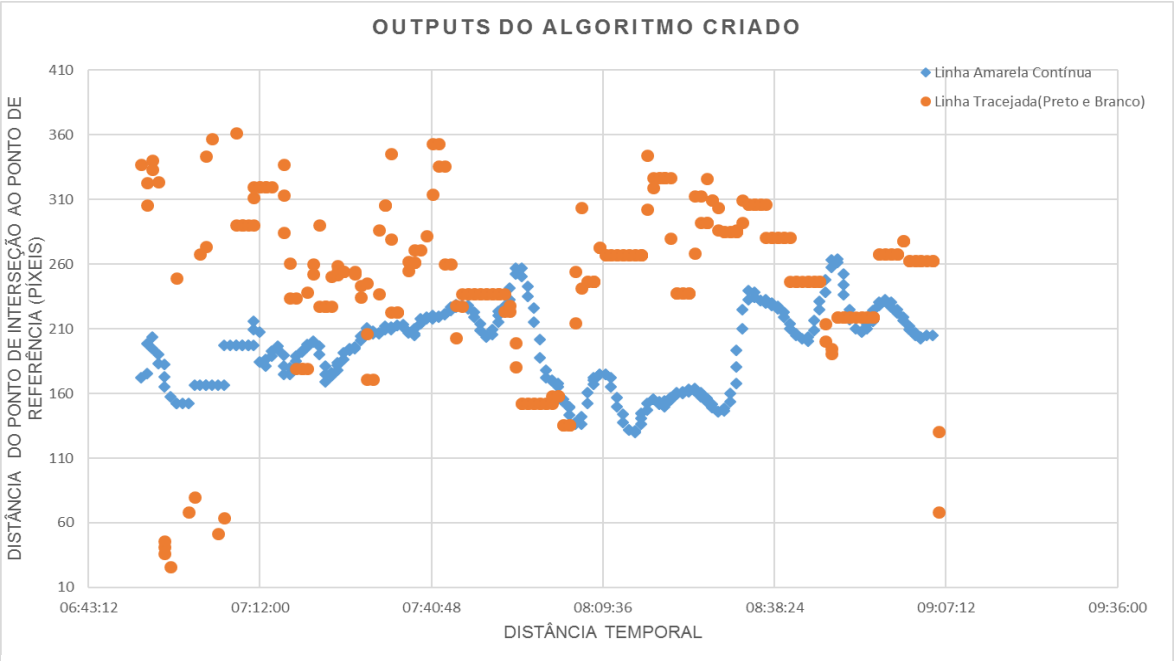


Figura 85 - Outputs do novo algoritmo ao longo de um percurso completo.

Pela análise da Figura 85, podemos verificar que os valores obtidos para P2 (a linha tracejada branca) apresentam uma grande variação de amplitude ao longo do percurso e uma maior dispersão, comparados com os valores de P1(linha amarela contínua). A dispersão das distâncias para os valores de P2, podem comprometer o desenvolvimento de um controlo de direção estável.

Analisando os dados da *Tabela 15*, a linha branca é detetada menos vezes uma vez que é uma linha descontínua. Os valores obtidos para a linha amarela contínua são mais estáveis tendo uma menor amplitude de variação em comparação com os valores para P2 (*Figura 84*), estando esta apta a servir de referência para um futuro controlador.

Número total de medições (por volta)	281
Nº de vezes que a linha amarela é detetada	281
Nº de vezes que a linha branca tracejada é detetada	263

Tabela 15 - Eficácia na deteção das linhas delimitadoras.

Uma vez que os valores de P2 são demasiado dispersos (sofrem grandes variações de amplitude aleatóriamente), propõe-se a colocação, do lado esquerdo do percurso, de uma linha amarela contínua, semelhante à linha que marca o percurso no lado direito. Assim, consegue-se

estabilizar os valores de P1 e P2 e partir para o desenvolvimento de um controlador de direção estável (sem grandes oscilações e o mais centrado possível com o centro da pista).

5.9. ANÁLISE DO CONSUMO DE ENERGIA AO LONGO DE UM TURNO

O controlo dos consumos de energia ao longo dos turnos é necessário, assim como a garantia de que o Segway tem autonomia para aguentar pelo menos um turno completo. Por isso, neste capítulo, faz-se a análise da variação do nível de energia ao longo de um turno.

Antes de apresentar os resultados, é importante referir que os testes para análise do consumo de bateria foram efectuados em condições normais de funcionamento, com três operadores a utilizar o Segway, por turno. O Segway foi ligado cerca de 30 min antes do ínicio do turno e parado 10 minutos antes do fim do turno. Os últimos 10 minutos são utilizados para a troca entre turnos, tempo este que é suficiente para realizar a troca de baterias, se necessária.

Pela análise dos gráficos das Figura *86 e* Figura *87,* podemos concluir que apesar de uma das baterias ter uma taxa de débito de energia superior à outra bateria (facto explicado na *sub-secção 4.7)* isto não tem qualquer impacto na autonomia final do Segway, que é mais que suficiente para fazer um turno completo de 8 horas. No entanto, os testes efectuados não são suficientes para quantificar ao certo a taxa de débito de energia que cada equipamento provoca em cada bateria do Segway. Durante um turno completo a percentagem de consumo de energia é de 62.5% da energia total inicial armazenada no conjunto das duas baterias.

É importante que, daqui para a frente, quando seja adicionado qualquer equipamento extra, que requeira consumo de energia das baterias do Segway, se meçam os seus consumos e se volte a verificar a autonomia do Segway.

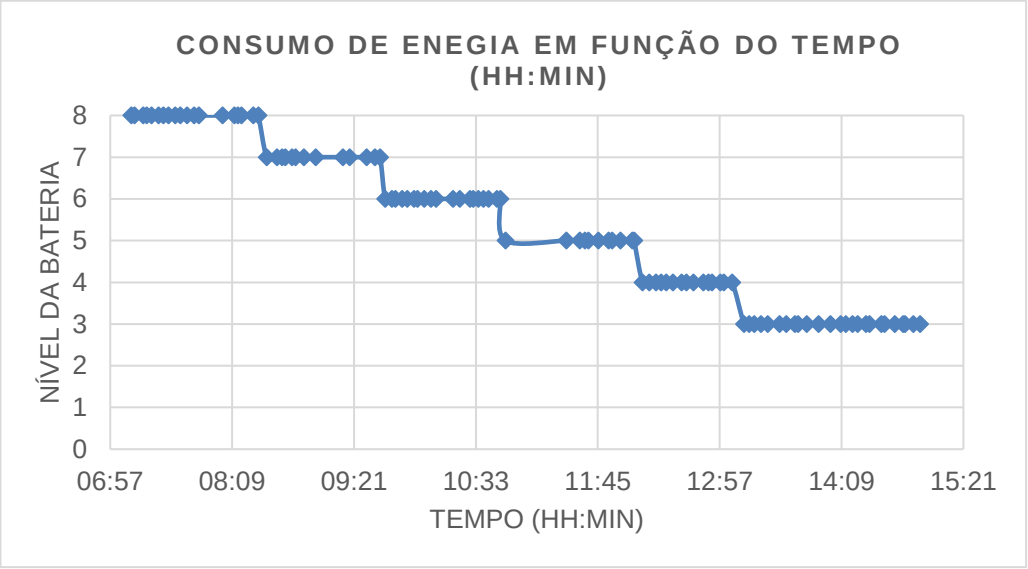


Figura 86 - Variação do nível de bateria por tempo.

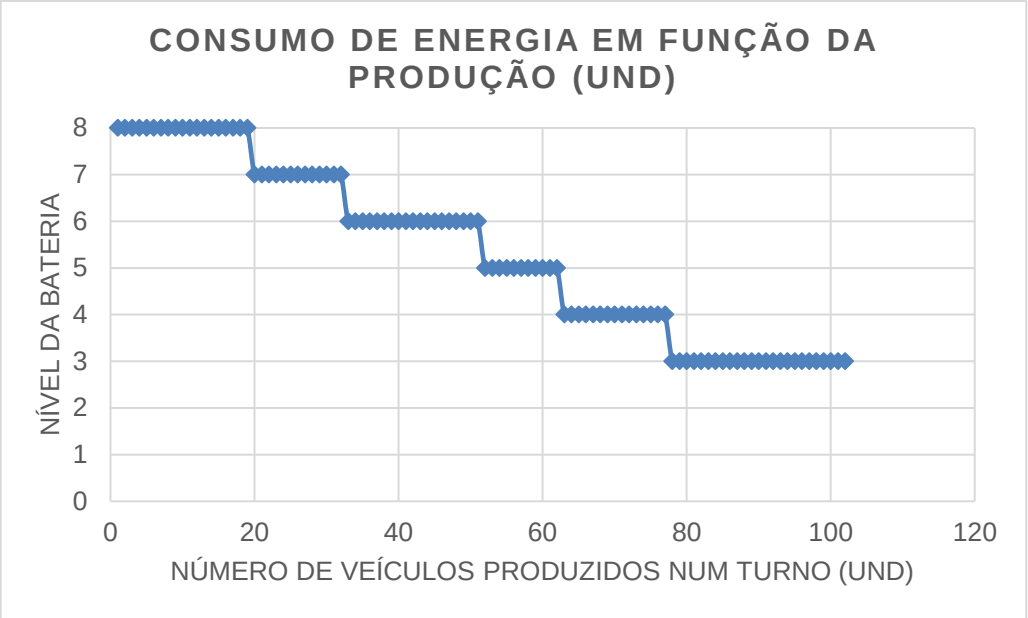


Figura 87 - Variação do nível de bateria por número de véiculos produzidos.

5.10. ENSAIOS GERAIS

Na primeira fase do estágio realizaram-se três ensaios, nos três turnos que operam na fábrica (manhã, tarde e noite), a fim de se quantificarem os problemas que foram abordados no capítulo quatro. Analisando o gráfico da Figura 88, podemos verificar que os problemas provocados pela iluminação são os que têm maior relevo, secundado pelos do algoritmo, nomeadamente, os valores HSV com que a linha amarela é segmentada; a região de interesse utilizada e o algoritmo para o controlo de direção. Por último, mas não menos significativos, temos os que surgem da não

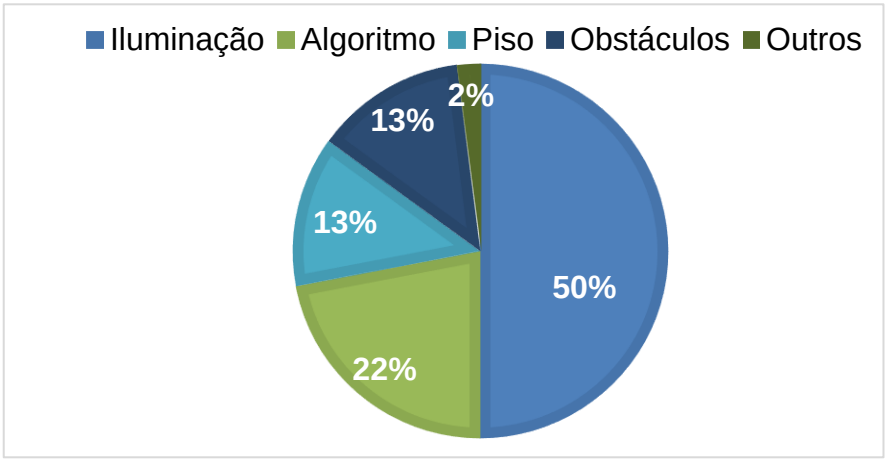


Figura 88 - Quantificação dos problemas.

deteção de obstáculos/objetos - o veículo não tinha um sensor de distância implementado - e os problemas provocados pelo mau estado do piso.

Ao longo do estágio realizaram-se ensaios para quantificar o aumento de desempenho e rendimento operacional global que o Segway foi tendo com as soluções apresentadas no capítulo cinco. Os ensaios foram feitos durante turnos completos de oito horas em condições normais de utilização por parte dos utilizadores (não houve ausências) e de produção (produção superior a 80% face aos valores médios diários). Para registar os valores das medições ao longo dos ensaios foi utilizada uma tabela (ANEXO 6).

Por cada veículo que entra no controlo de qualidade (BTU), o Segway faz um percurso completo (ida em modo manual e volta em modo autónomo). O número total de medições por ensaios realizados foi tido em conta, por se considerarem os suficientes, para tornarem válidos e representativos os resultados dos mesmos. O valor de performance (*Equação 21*) foi calculado com base no número total dos percursos completos, desde o ponto inicial até ao ponto de chegada, sem qualquer falha (quando o Segway não parou ou saiu da sua trajetória normal e conseguiu chegar ao ponto final, sem que houvesse intervenção de um operador).

Na Tabela 16, podemos ver as performances obtidas ao longo dos diferentes ensaios que se realizaram durante o estágio. Os três primeiros ensaios, representam o estado em que o Segway se encontrava inicialmente.

Deste o ensaio 4 até ao ensaio 8, foram feitas alterações de modo a calcular o impacto que cada uma tinha no final de cada turno. Sempre que se obtinha uma melhoria na performance, as alterações feitas eram mantidas no ensaio seguinte, ou seja, no ensaio seguinte, as alterações do ensaio anterior estavam à partida implícitas.

$$Performance = \frac{nº\ de\ percusos\ sem\ falhas}{nº\ total\ de\ percursos} \quad (21)$$

Ensaio	Turno	Desempenho (%)	Nº de medições	Alterações Mais significativas	Objetivo
1	Manhã	**53.3**	73	--	Perceber os principais problemas existentes no sistema e avaliação das potênciais causas e consequências
2	Tarde	**22.7**	88	--	Perceber os principais problemas existentes no sistema e avaliação das potênciais causas e consequências
3	Noite	**2**	49	--	Perceber os principais problemas existentes no sistema e avaliação das potênciais causas e consequências
4	Manhã	**69.6**	102	Novos valores HSV Aumento da região de interesse Alteração no controlador de direção	Perceber o impacto das alterações na performance final. Medir níveis do consumo da bateria
5	Tarde	**51.9**	27	Com Holofote e RPLidar	Testar impacto de ter iluminação permanente
6	Manhã	**76.5**	46	Linha Nova Sem Holofote	Testar o impacto da nova linha
7	Noite	**20**	48	Sem Holofote Alterações no controlador Nova linha amarela	Testar o impacto da nova linha no algoritmo
8	Tarde	**89.5**	92	Alteração da posição da câmara	Testar o impacto da iluminação e consequentes reflexos com uma posição da câmara diferente.
9	Noite	**75**	24	Alteração da posição da câmara	(Igual ao ensaio 8)

Tabela 16 - Performances Parciais.

Depois de analisados os resultados ao longo dos vários ensaios da tabela (*Tabela 16)* podem ser retiradas algumas conclusões importantes. Com as melhorias no algoritmo de controlo de direção, o Segway desloca-se mais próximo do centro da trajetória e com menor amplitude das suas oscilações. Como consequência, o Segway acabou por diminuir significativamente o número de vezes que saía dos limites (delimitados pela linha tracejado branca e linha contínua amarela), facto que pode ser verificado pelos dados da Figura 89, onde se faz a comparação entre o "antes" e o "depois" das alterações no algoritmo de controlo de direção.

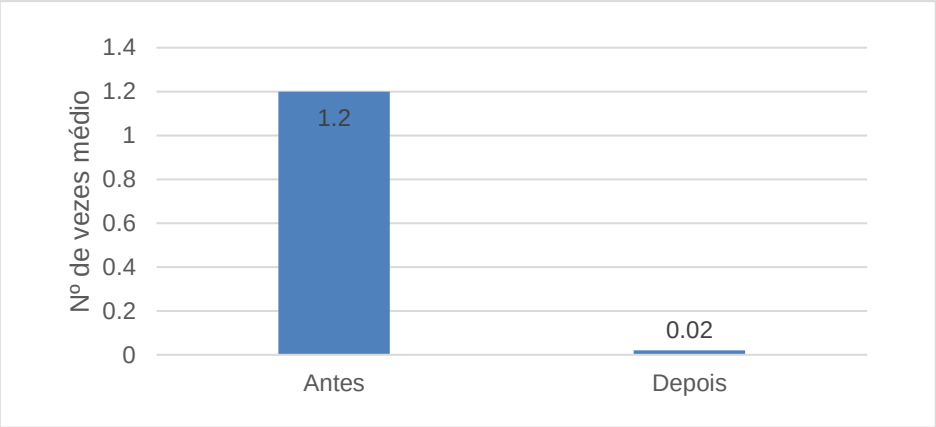


Figura 89 - Número de vezes médio que o Segway sai dos limites do percurso.

Esta melhoria no controlo de direção permite diminuir o tempo que o Segway demora a realizar um trajeto completo, para dois minutos e seis segundos. Contudo, este tempo de ciclo tende a aumentar, caso haja alguma falha e esta não seja rapidamente solucionada, para tempos no mínimo, superiores a dois minutos e trinta e cinco segundos (*Figura 90).*

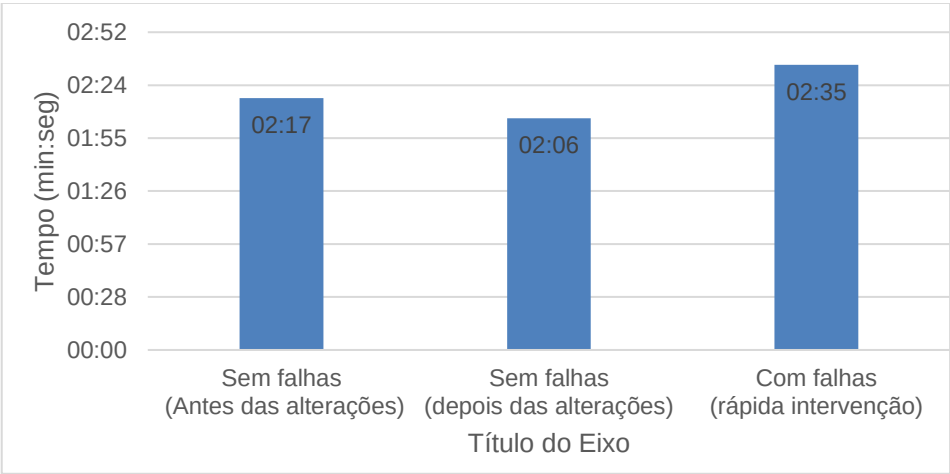


Figura 90 - Tempo médio de uma volta (MM:SS)

Os números, da *Figura 89*, induzem um aumento significativo no tempo médio entre falhas (TMBF) uma vez que, saindo menos vezes dos limites do percurso, o Segway poderá completar o circuito sem falhas.

Na *Figura 91*, *Figura 92 e Figura 93* estão representadas as evoluções que foram feitas ao longo dos ensaios. Pela análise das mesmas, concluímos que a performance do Segway aumentou em cerca de 23,2% no turno da manhã, 66.8 % no turno da tarde e 73% no turno da noite. Durante o turno da manhã os principais fatores que contribuiram para o aumento de desempenho foram a melhoria das condições do piso e a melhoria no algoritmo de direção (possíbilitando-se assim um controlo mais estável, com menos oscilações, menos amplitudes e mais próximo do centro da pista), com um impacto equivalente a 30% e 60% respetivamente. Durante o turno da tarde e o turno da noite a alteração que teve maior impacto para o aumento do desempenho foi o reposicionamento da câmara (em altura e inclinação) com um valor de impacto que ronda os 60% e 75% respetivamente.

Faltou, no entanto, testar o impacto do filtro polarizador ao longo dos três turnos, uma vez que o suporte para a câmara não está apto a ser utilizado juntamente com o filtro polarizador. No entanto, fica provado, no capítulo cinco, que o filtro polarizador tem um impacto significativo no aumento dos contrastes entre a linha e o piso, o que pode ser importante para alcançar melhores valores de performance.

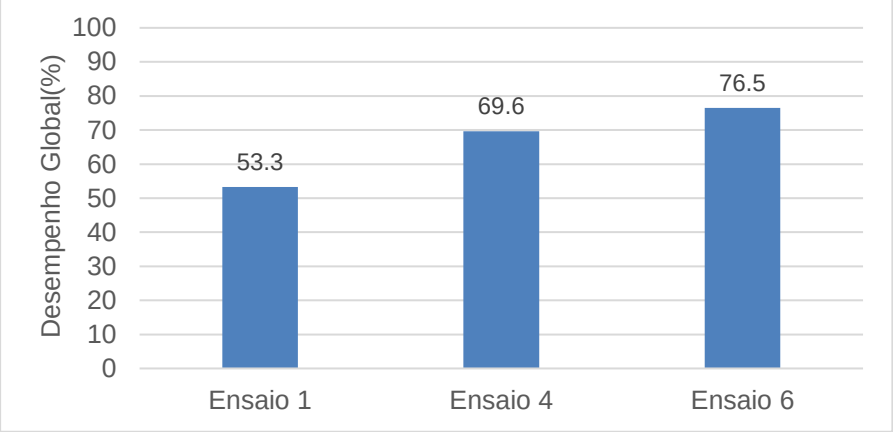


Figura 91 - Ensaios no turno da manhã.

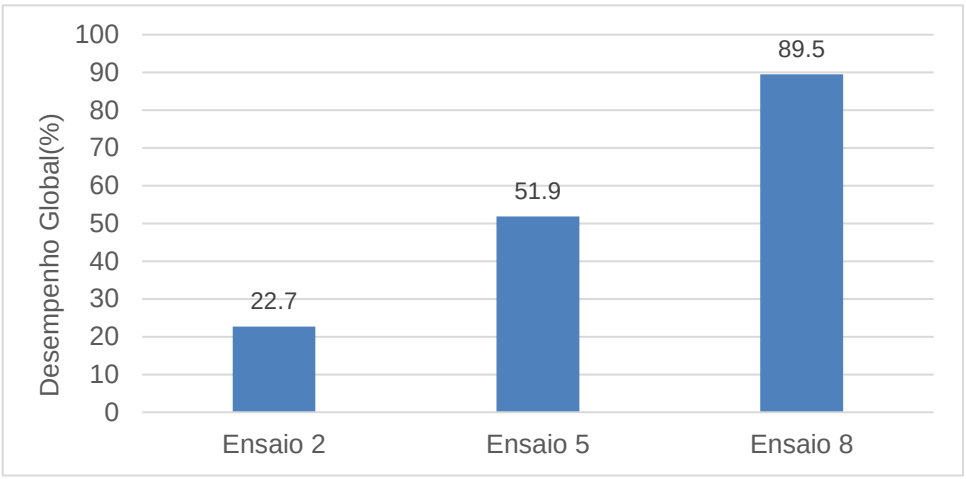


Figura 92 - Ensaios no turno da tarde.

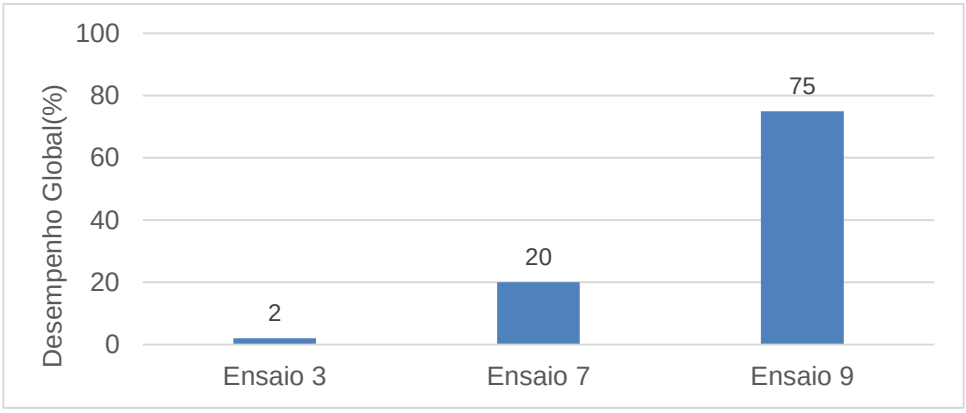


Figura 93 - Ensaios no turno da noite.

Com as melhorias efetuadas foi possível aumentar para 80% o rendimento operacional do Segway. Em comparação com os 26% de rendimento operacional que o equipamento tinha inicialmente, consegui-se um aumento que ronda os 54%. Para este aumento, são ilustradas na Figura *94*, as alterações que mais contribuíram.

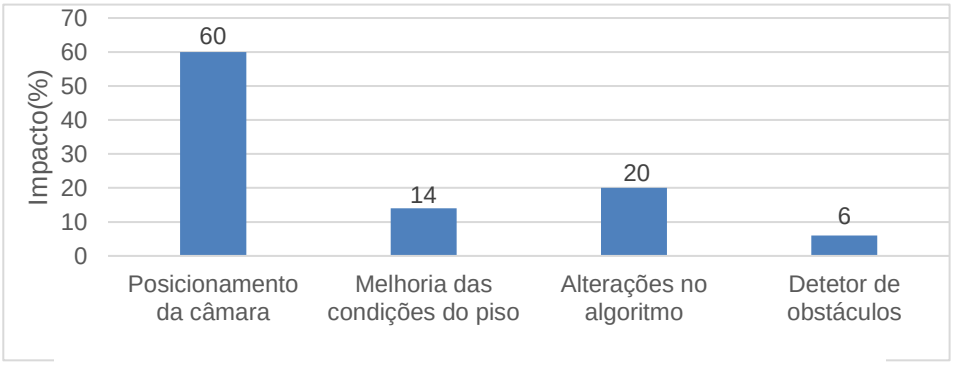


Figura 94 - Impacto em % das alterações, no aumento do rendimento operacional.

Com as alterações apresentadas no capítulo quatro, neste momento os únicos problemas que afetam o desempenho do Segway são:

- O sensor de distância, uma vez que o sensor ótico utilizado sofre interferências do meio e é sensivel ao tipo e cor das superficies dos objetos.
- O piso, dado que a fita adesiva amarela tem tendência a se degradar e já se encontra de novo com marcas devido à frequente passagem de veículos.

Na Figura *95* estão quantificados estes mesmos problemas.

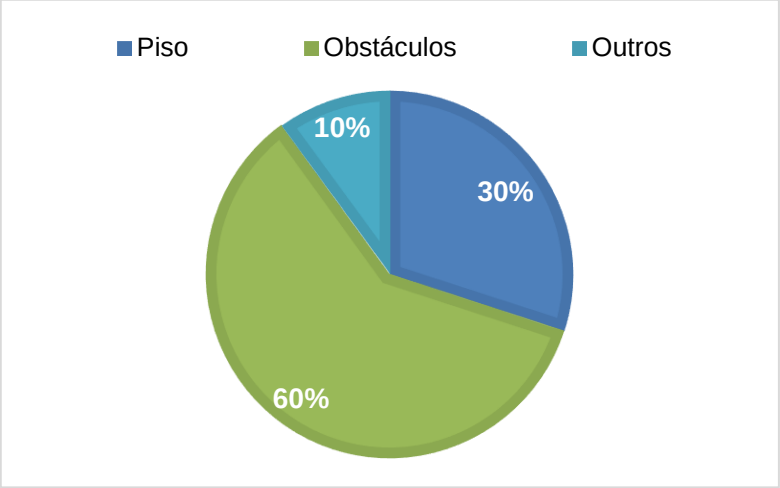


Figura 95 - Desafios atuais

6. CONCLUSÃO E TRABALHOS FUTUROS

No presente capítulo serão abordadas as principais conclusões que se retiram deste estágio e será dado mote para trabalhos futuros que se poderão realizar. Será também feito uma pequena avaliação pessoal sobre dificuldades que tiveram de ser ultrapassadas.

No fim de implementadas as alterações e soluções propostas, o Segway está atualmente com um rendimento operacional de 80%. Ou seja, ao longo de um turno completo, em 100 carros, o Segway consegue fazer 80 vezes o seu percurso sem falhas. No entanto ainda não é suficiente para estar funcional.

Embora se tenha melhorado o desempenho do Segway, o algoritmo antigo continuava sensível a certas situações (reflexos amarelos no piso e degradação do mesmo). Com o desenvolvimento de um novo algoritmo baseado na deteção das linhas que delimitam o trajeto, os problemas, como os reflexos amarelos dos carros e de algumas lâmpadas led, coletes amarelos, deixam de ter qualquer impacto, provando uma vez mais que este algoritmo tem um enorme potencial, podendo aumentar em cerca de 3% o rendimento operacional final do Segway.

Um sistema de deteção de obstáculos foi implementado, no entanto, não demonstrou ser suficiente apto a detetar a maioria dos objetos. Por isso, é necessário encontrar alternativas. Estima-se que com um equipamento de deteção de obstáculos robusto, o desempenho operacional aumente até 10%.

Em relação ao tempo médio entre falhas (*TMBF*), este aumentou significativamente com as alterações efetuadas e tende a aumentar se as operações de manutenção e regras de manobra do Segway forem respeitadas, que corresponde a um aumento de 5% no rendimento operacional.

É importante também revelar que durante este estágio realizou-se um registo da maior parte dos fatores que podiam afetar a condução autónoma do Segway, nomeadamente, o seu impacto quantitativo e qualitativo no desempenho final, algo que, até então, ainda não estava bem claro nem documentado. Para evitar acidentes futuros, foi elaborado um conjunto de regras que são importantes e terão de ser aplicadas por parte dos operadores da BTU, para que acidentes, durante a operação do Segway, não voltem a acontecer, nomeadamente, as quedas por falta de energia nas baterias do segway ou por excesso de velocidade, quando o piso está molhado e escorregadio.

Para concluir, desenvolveu-se um vídeo onde se resume em poucos minutos algum do trabalho feito neste estágio. O vídeo encontra-se disponível no link[27] em rodapé.

A nível de dificuldades sentidas, o tempo para receção das encomendas do material necessário foi o maior fator crítico, uma vez que retardou o trabalho desenvolvido.

Embora estejam cumpridos todos os objetivos propostos para este estágio, se continuarmos a seguir a metodologia de um ciclo *PDCA,* é importante traçar novos objetivos face aos resultados e conclusões que se obtiveram no fim deste primeiro ciclo. Propõem-se, por isso, um leque de trabalhos futuros a realizar:

[27] Link para o vídeo: https://youtu.be/KkmXYOkNKQw , plataforma *Youtube*
Link para *Google Drive*: https://drive.google.com/open?id=1-FQtJGygTUp494lw0qQMiuEBbzgTNkdu

- Em alternativa ao sensor ótico (para ambientes interiores) usado, propõe-se a implementação de um sensor ultra-sons [28], ou um sensor ótico para ambientes exteriores, como sensores de distância.
- Implementar um algoritmo de controlo de direção e navegação estável para complementar o algoritmo das linhas virtuais do Segway.
- Evoluir o algoritmo de deteção de linhas virtuais retas, para um algoritmo capaz de detetar linhas virtuais curvas.
- Avaliar a performance do novo algoritmo e comparar os resultados com os valores obtidos neste relatório.
- Pensar numa maneira de comercializar o projeto "*Segway*", de modo a exportar este prótotipo para todas as fábricas do *Grupo PSA*.
- Desenvolver uma análise quantitativa aos efeitos dos consumos efetuados por cada equipamento em cada uma das baterias do Segway.
- Pôr em prática medidas para manter o piso seco e em bom estado.

Estima-se que com algumas destas alterações o Segway aumente significativamente o seu rendimento operacional. Ora, tendo em conta o rendimento operacional alcançado durante a realização deste estágio e o sucesso destas alterações prevê-se que o Segway funcione plenamente e sem falhas. Poderão ser realizadas ainda, melhorias ao nível do equipamento mecânico e hardware utilizado no sistema para aumentar a capacidade de processamento e reação de alguns equipamentos (processadores, controladores e atuadores).

[28] Module HC - SR04, "Ultrasonic Ranging Module HC - SR04 datasheet".

7. REFERÊNCIAS

[1] A. Rodrigues, “Proposta de ações de melhoria no setor da Qualidade: um caso de estudo na indústria automóvel”, Coimbra (Portugal), 2017.

[2] B. Browning, “Turning Segways into soccer robots”, em *Industrial Robot: An International Journal,* vol. 32, nº 2, pp. 149-156, 2005.

[3] S. Hojjat, “Region Growing: A New Approach”, Fevereiro de 1998.

[4] Lego Group, “LEGO MINDSTORMS EDUCATION, NXT User Guide”, 2006.

[5] E.Wijaya Kurniawan, “Segway Line Tracer Using Proportional-Integral-Derivative Controllers”, Brawijaya (Indonésia), 2016.

[6] Gonçalves, José Carlos da Costa, “Visão Artificial em Condução Autónoma com Câmara Kinect”, Universidade do Minho (Portugal), 2013.

[7] S. Zennaro, “Evaluation of Microsoft Kinect 360 and Microsoft Kinect One for robotics and computer vision applications”, Padova (Itália), 2014.

[8] Q. Zou, “Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks”, Wuhan University (China), 2019.

[9] H. Chen, “Robust Lane Detection for Complicated Road Environment Based on Normal Map”, Shandong University,Jinan (China), 2018.

[10] A. Nascimento, “Comparação e classificação de técnicas de estereoscopia para realidade aumentada e jogos”, São Paulo (Brazil), 2010.

[11] P. Craig Stark, “Debayering Demystified”, em *AstroPhoto Insight*.

[12] National Intruments, “A Practical Guide to Machine Vision Lighting”, 19 Março 2019. Disponível online em: http://www.ni.com/fr-fr/innovations/white-papers/12/a-practical-guide-to-machine-vision-lighting.html.

[13] K. Kaur, “Performance evaluation of dbla,hsv-clahe & rgb-clahe based image enhancement”, em *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),* 2014.

[14] Ghassan Hamarneh,Karin Althoff,Rafeef Abu-Gharbieh, “Automatic Line Detection”, Chalmers University, Lund, 1999.

[15] W. A. ADAMS, “Analysis of Robustness in Lane Detection using Machine Learning Models”, Ohio University (United States), 2015.

[16] M. M. Clothier, “Overcoming Difficulties of Tracking in Changing Lighting Conditions”, University of California, San Diego, 2004.

[17] A. Owen-Hill, “Robot Vision Lighting: Why There's No Perfect Setup”, 1 Setembro 2016.

[18] A. Oliveira, “Fundamentos de Controle de Processo”, CPM-Programa de Certificação de Pessoal de Manutenção, 1999.

[19] Autor Desconhecido, “Cómo funciona un control proporcional derivativo?”,Universidade de Manizales(Colômbia), 2011.

[20] J. Alves, “Navegação Híbrida num Veícculo Autónomo com Direção Ackermann”, Universidade de Aveiro (Portugal), 2016.

[21] S. Skogestad, “Probably the best simple PID tuning rules in the world”,em *Journal of Process Control,* 2001.

[22] M. Kumano, “Obstacle Avoidance of Autonomous Mobile Robot using Stereo Vision Sensor”, Ibaraki (Japan), 2000.

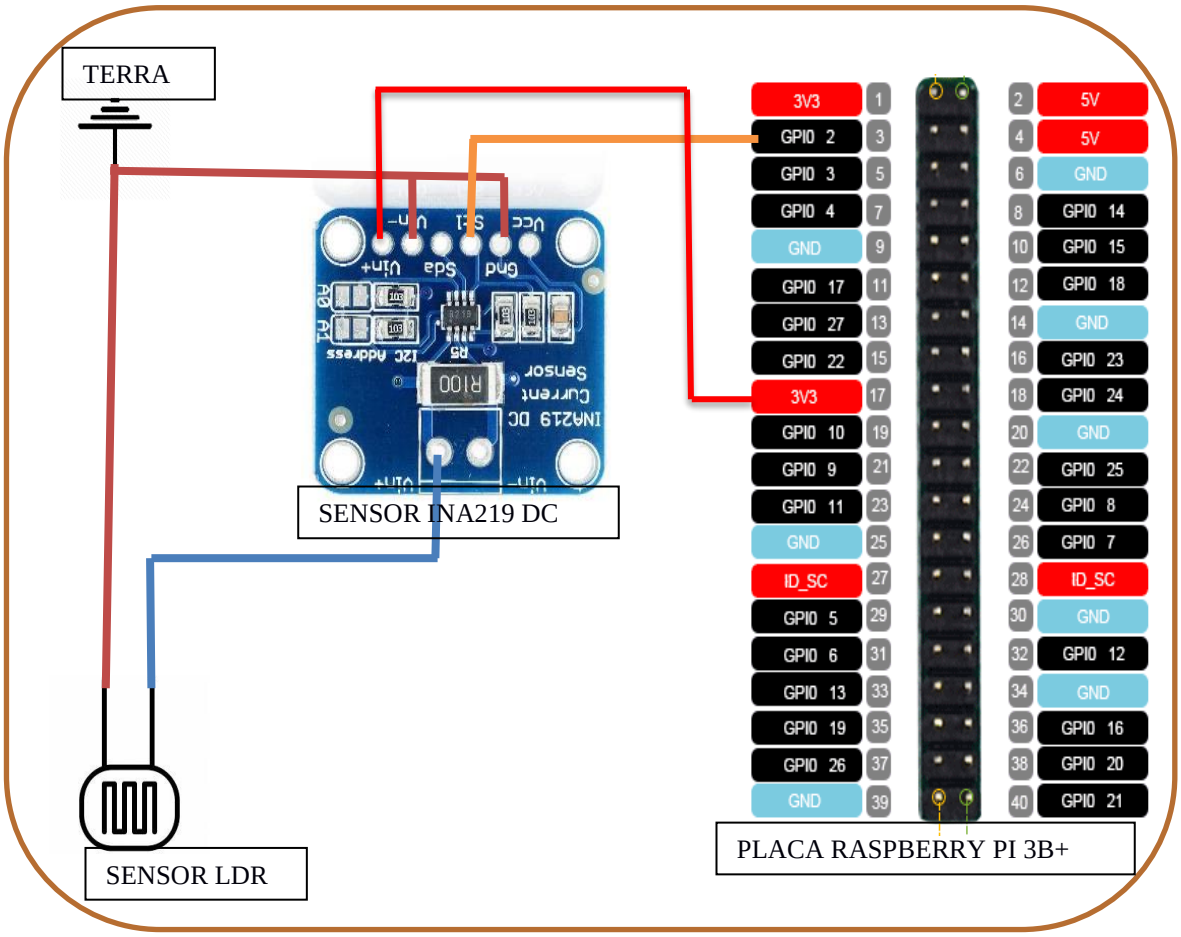
[23] D. Silva, “Calibração Multissensorial e Fusão de Dados Utilizando LIDAR e Visão”, Universidade de Aveiro (Portugal), 2016.

[24] A. Martínez Tenor e Fernández Madrigal, “Lego Mindstorms NXT and Q-learning: a teaching approach for robotics and engineering”, Spain, 2014.

ANEXOS

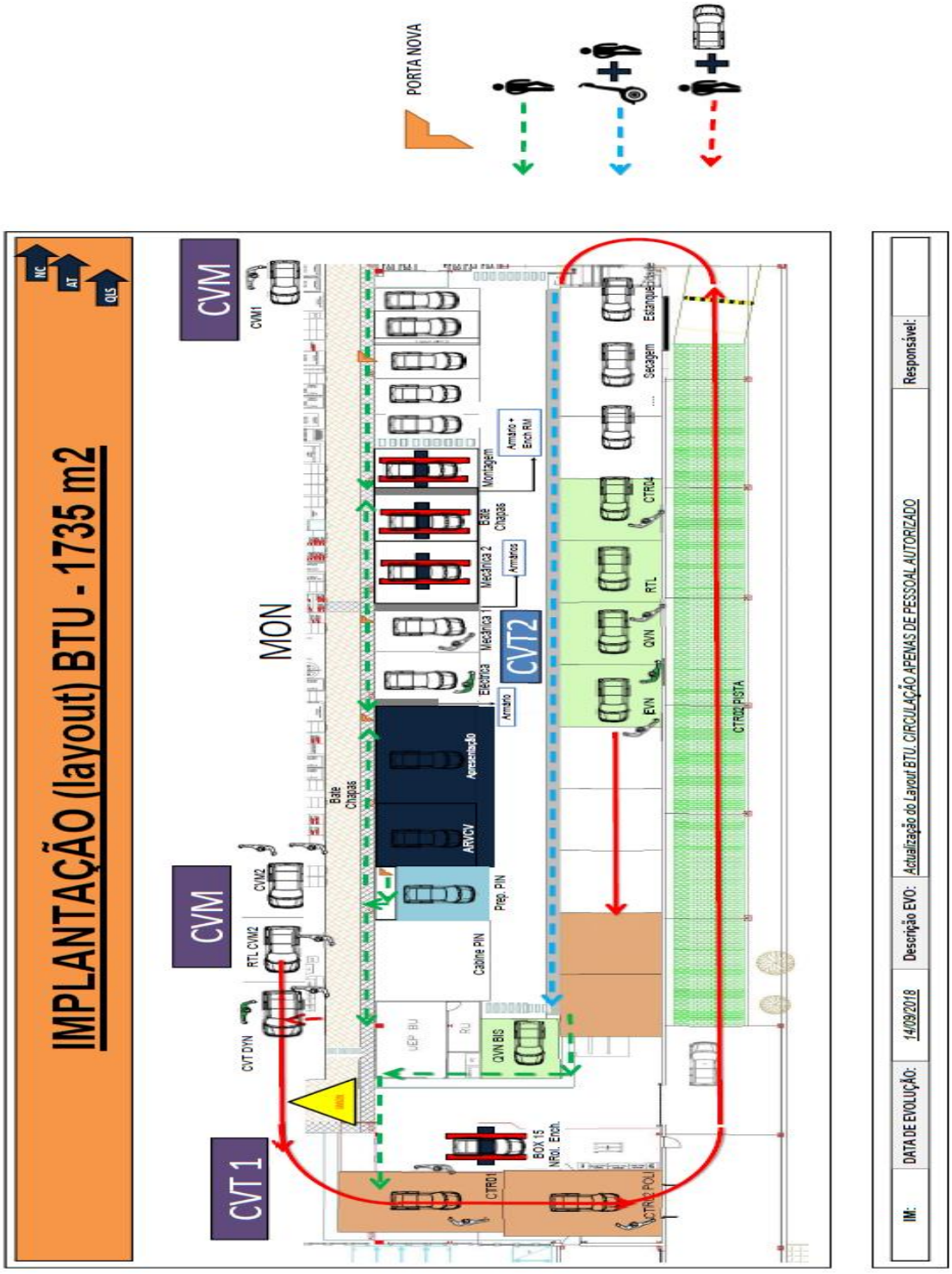
ANEXO I

(CIRCUITO ELÉTRÓNICO PARA MEDIR VARIAÇÕES DE LUMINOSIDADE)



ANEXO 2

(MAPA DE CIRCULAÇÃO NO BTU)



ANEXO 3

(ALGORITMO PARA DETEÇÃO DE LINHA)

```
#função para detetar as arestas da linha branca
def detect_edges_White(image, low_threshold=100, high_threshold=200):
  #aconselhado usar uma proporção de 1:3 ou 1:2 para low_threshold:high_threshold
  return cv2.Canny(image, low_threshold, high_threshold)

#função para detetar as arestas da linha amarela
def detect_edges(image, low_threshold=200, high_threshold=400):
  return cv2.Canny(image, low_threshold, high_threshold)

#função para detetar linhas numa imagem binarizada
def hough_lines(cannyImage):
   lines= cv2.HoughLinesP(cannyImage, rho=1,theta= np.pi/180, threshold=100,
minLineLength=80, maxLineGap=3)
   return lines

#função que deteta a linha tracejada no lado esquerdo
def drawWhiteLine(img,lines):
   global rightInterceptX,rightInterceptY, slope, VLx, VLy, buffer1,buffer2,buffer3, estadoL,m,
VLxmean, VLymean
   estadoL=0
   rightInterceptY=[]
   rightInterceptX=[]
   VLx=[]
   VLy=[]
   width = img.shape[1]
   height = img.shape[0]
   try:
      for x in range(0,len(lines)):
         for x1,y1,x2,y2 in lines[x]: #x1,y1,x2,y2 são os extremos de cada reta encontrada
devolvidos pela função HoughLinesP e que foram armazenados num buffer “lines”.
            if x2 != x1:
               slope= ((y2-y1)/(x2-x1))
               if slope < -0.4:
                  interceptY = y1 - (slope*x1)
                  interceptX = (-interceptY)/slope
                  rightInterceptX.append(interceptX)
                  rightInterceptY.append(interceptY)
            else:
               slope=0
      mediaX=np.average(rightInterceptX)
      mediaY=np.average(rightInterceptY)

      #plot dos pontos
      ‘’’
      plt.plot(rightInterceptX,rightInterceptY,'ro',mediaX,mediaY,'g^')
      plt.ylabel('Y interception')
      plt.xlabel('X interception')
      plt.show()
      plt.savefig('plote.png')
      ‘’’
      if len(lines) > 0: #é importante que sejam detetadas retas
         for x in range(0,len(lines)):
            for x1,y1,x2,y2 in lines[x]:
```

```
                if x2 != x1:
                    slope= ((y2-y1)/(x2-x1))
                    if slope < -0.4:
                        cv2.line(img,(x1,y1),(x2,y2),(100,255,0),2) #desenha a linha
                        interceptY = y1 - (slope*x1)
                        interceptX = (-interceptY)/slope
                        distancia=(np.sqrt((interceptX-mediaX)**2 + (interceptY-mediaY)**2))
                        if distancia < 5000:
                            cv2.line(img,(x1,y1),(x2,y2),(0,255,0),2)
                            VLx.append(interceptX)
                            VLy.append(interceptY)
                else:
                    slope=0
            #vamos calcular media das retas escolhidas (filtro de ruído)
            VLxmean=np.average(VLx)
            VLymean=np.average(VLy)
            m= ((float(-int(VLymean))/(int(VLxmean))))
            if VLxmean != 0:
                buffer1.append(VLxmean)
                buffer2.append(VLymean)
                buffer3.append(m)

            if len(buffer1) > 7: #aqui escolhemos o número total de frames anteriores que queremos
utilizar para calcualr a média
                try:
                    VLxmean=int(np.mean(buffer1[-5:]))
                    VLymean=int(np.mean(buffer2[-5:]))
                    m=int(np.mean(buffer3[-5:]))
                except (ValueError, TypeError, ZeroDivisionError, OverflowError, RuntimeWarning):
                    print('Aconteceu um erro')
                    VLxmean=0
                    VLymean=0
                    m=0
            cv2.line(img,(0,int(VLymean)),(int(VLxmean),0),(100,255,100),8) #desenha linha virtual
representativa
            #Vamos calculara distancia do centro da imagem à linha
            Xref=int(0.5*width)
            Yref=int(0.7*height)
            if VLxmean !=0 and m != 0:
                distancia= np.sqrt((Xref-VLxmean)**2)
                estadoL=1
            else:
                estadoL=0
                distancia=0
            #vamos determinar x1 e y1 que sao os pontos da reta
        else:
            print('No line')
            VLxmean=0
            VLymean=0
            m=0
            estadoL=0
            distancia=0
    except (ValueError, RuntimeWarning):
        estadoL=0
        distancia=0
    return (estadoL,m,VLymean, VLxmean, distancia)
```

```
#função que desenha a linha virtual sobre a linha amarela

def drawRightLine(img,lines):
    global rightInterceptX,rightInterceptY, slope, VLx, VLy, buffer1,buffer2,buffer3, estadoR, m,
VLxmean, VLymean
    estadoR=0
    rightInterceptY=[]
    rightInterceptX=[]
    VLx=[]
    VLy=[]
    width = img.shape[1]
    height = img.shape[0]
    try:
        for x in range(0,len(lines)):
            for x1,y1,x2,y2 in lines[x]:
                if x2 != x1:
                    slope= ((y2-y1)/(x2-x1))
                    if slope > 0.6:
                        cv2.line(img,(x1,y1),(x2,y2),(255,255,0),2)
                        interceptY = y1 - (slope*x1)
                        interceptX = (height-interceptY)/slope
                        rightInterceptX.append(interceptX)
                        rightInterceptY.append(interceptY)
                else:
                    slope=0
        mediaX=np.average(rightInterceptX)
        mediaY=np.average(rightInterceptY)
        '''
        plt.plot(rightInterceptX,rightInterceptY,'ro',mediaX,mediaY,'g^')
        plt.ylabel('Y interception')
        plt.xlabel('X interception')
        plt.show()
        plt.savefig('plote.png')
        '''
        if len(lines) > 0:
            for x in range(0,len(lines)):
                for x1,y1,x2,y2 in lines[x]:
                    if x2 > x1 or x1 > x2:
                        slope= ((y2-y1)/(x2-x1))
                        if slope > 0.6:
                            cv2.line(img,(x1,y1),(x2,y2),(255,255,0),2)
                            interceptY = y1 - (slope*x1)
                            interceptX = (height-interceptY)/slope
                            distancia=(np.sqrt((interceptX-mediaX)**2 + (interceptY-mediaY)**2))
                            if distancia < 800:
                                cv2.line(img,(x1,y1),(x2,y2),(0,255,0),2)
                                VLx.append(interceptX)
                                VLy.append(interceptY)
                    else:
                        slope=0
            VLxmean=np.average(VLx)
            VLymean=np.average(VLy)
            m= (((height-int(VLymean))/(int(VLxmean))))
            print('VLxmeanR=',VLxmean)
            print('mR=',m)
            if VLxmean != 0:
                buffer1.append(VLxmean)
                buffer2.append(VLymean)
```

```
            buffer3.append(m)

        if len(buffer1) > 7:
            try:
                VLxmean=int(np.mean(buffer1[-5:]))
                VLymean=int(np.mean(buffer2[-5:]))
                m=int(np.mean(buffer3[-5:]))
            except (ValueError, TypeError, ZeroDivisionError, OverflowError, RuntimeWarning):
                print('Aconteceu um erro')
                VLxmean=0
                VLymean=0
                m=0
            cv2.line(img,(0,int(VLymean)),(int(VLxmean),height),(0,255,255),8)
            #Vamos calculara distancia do centro da imagem à linha
            Xref=int(0.5*width)
            Yref=int(0.7*height)
            if VLxmean !=0 and m != 0:
                distancia= np.sqrt((Xref-VLxmean)**2)
                print('distanciaR=',distancia)
                estadoR=1
            else:
                estadoR=0
                distancia=0
            #vamos determinar x1 e y1 que sao os pontos da reta
        else:
            print('No line')
            VLxmean=0
            VLymean=0
            m=0
            estadoR=0
            distancia=0
    except (ValueError, RuntimeWarning):
        estadoR=0
        distancia=0
    return (estadoR,m,VLymean, VLxmean, distancia)



#main
While True:
    try:
        #ler as imagens
        img = cv2.imread(f1)
        #ler dados todos do ficheiro pickle que guarda as constantes
        height=img.shape[0]
        width=img.shape[1]
        hh = cv2.getTrackbarPos('H high', 'Image Settings')
        sh = cv2.getTrackbarPos('S high', 'Image Settings')
        lh = cv2.getTrackbarPos('V high', 'Image Settings')
        hl = cv2.getTrackbarPos('H low', 'Image Settings')
        sl = cv2.getTrackbarPos('S low', 'Image Settings')
        ll = cv2.getTrackbarPos('V low', 'Image Settings')
        disp = cv2.getTrackbarPos('Displacement', 'Image Settings')
        frame = np.array(img,copy=True)
        frame2 = np.array(img,copy=True)
        frame3= np.array(img,copy=True)
        #filtrar os amarelos e os brancos da imagem
```

```
        white_yellow_images = select_white_yellow (frame2) #COM OU SEM CAHLE
        #cv2.imshow('white_yellow', white_yellow_images )
        #converter os brancos para tons cinza
        gray_images_white = convert_gray_scale(frame)
        #cv2.imshow('gray_image', gray_images)
        #vamos detetar arestas
        edges_white =detect_edges_White(gray_images_white)
        cv2.imshow('edges_black_and_white', edges_white)
        #vamos defenir a regiao de interesse
        edges_white_croped=select_region_white(edges_white)
        #cv2.imshow('edges_black_and_white_croped', edges_white_croped)
        #filtro de ruido
        blurred_images_white = apply_smoothing (edges_white_croped)
        #cv2.imshow('edges_black_and_white_blurred', blurred_images_white)
        #vamos encontrar as linhas
        white_lines = hough_lines_white(blurred_images_white)
        #vamos desenhar as linhas na imagem
        (estadoL,mL,VLymeanL,VLxmeanL, distanciaL)=drawWhiteLine(frame3,white_lines)
        #cv2.imshow('blurred_images', blurred_images)
        #blurred_images=select_region(gray_images)
        #detetc edges
        #edge_images = detect_edges(gray_images)
        edge_images2 = detect_edges(white_yellow_images)
        #edge_images= cv2.dilate(edge_images, None, iterations=1)
        #edge_images = select_region(edge_images)
        edge_images2 = select_region2(edge_images2)
        #blurred_images = apply_smoothing (edge_images)
        blurred_images2 = apply_smoothing (edge_images2)
        #cv2.imshow('edge_images',blurred_images2)
        #aqui podes arranjar espaço para detetar linhas na imagem com o hough lines
        lines2 = hough_lines(blurred_images2)
        (estadoR,mR,VLymeanR, VLxmeanR, distanciaR)=drawRightLine(frame3,lines2)
        #cv2.imshow('amarelos',white_yellow_images)
        #cv2.imshow('contornos',edge_images2 )
        if estadoR == 1 and estadoL ==1 :
            print('MOVE ON!')
            cv2.putText(frame3,'MOVE ON', (int(width/2),int(height/2)),
cv2.FONT_HERSHEY_SIMPLEX , 1, (0,0,0), 2, cv2.LINE_AA)
            try:
                #podemos calcular a area assim
                #vamos encontrar os pontos
                print(VLymeanR)
                print(VLymeanL)
                print(mR)
                print(mL)
                x3=(float(VLymeanR-VLymeanL)/(mL-mR))
                y3=mL*x3 + VLymeanL
                print(x3,y3)
                base= np.absolute(distanciaL+distanciaR)
                print('Base=',base)
                altura=np.absolute(y3-height)
                print('alt=',altura)
                Area=((base)*altura)/2
                print('Area=',Area)
                cv2.putText(frame3,"Area=" + str(int(Area)), (int(width/2)+30,int(height/2)+30),
cv2.FONT_HERSHEY_SIMPLEX , 1, (0,0,0), 2, cv2.LINE_AA)
            except ZeroDivisionError:
                print('Error')
```

```
        if estadoR ==1 and estadoL ==0:
            print('Warning: Look to your distanceR')
            cv2.putText(frame3,'Warning: Look to your distanceR', (int(width/2),int(height/2)),
cv2.FONT_HERSHEY_SIMPLEX , 1, (255,0,0), 2, cv2.LINE_AA)
        if estadoR == 0 and estadoL ==1:
            print('Warning: Look to yout distanceL')
            cv2.putText(frame3,'Warning: Look to yout distanceL', (int(width/2),int(height/2)),
cv2.FONT_HERSHEY_SIMPLEX , 1, (0,255,0), 2, cv2.LINE_AA)
        if estadoR == 0 and estadoL ==0:
            print('STOP')
            cv2.putText(frame3,'STOP', (int(width/2),int(height/2)), cv2.FONT_HERSHEY_SIMPLEX ,
4, (0,0,255), 8, cv2.LINE_AA)
        cv2.imshow('final',frame3)
        k = cv2.waitKey(0)
        if(k==27):#ESC button
            exit()
        if k == ord("s"):
            with open('file.pickle','wb') as handle:
                pickle.dump({'REF X':px1,'REF Y':py1,'Y REC':y1,'W. Y1': ymin, 'W. Y2':ymax,'H.
X1':xmin,'H. X2':xmax, 'hl':hl, 'sl':sl, 'll':ll, 'hh':hh, 'sh':sh, 'lh':lh }, handle ,
protocol=pickle.HIGHEST_PROTOCOL)
                print ('gravei')
    except AttributeError:
        pass
```

ANEXO 4

(ALGORITMO PARA DETEÇÃO DE OBSTÁCULOS)

```
def detectObstacles():
	global ons1, ons2, andamento, modo_livre, sentido, killThreads, freeMode, move, dista
	while not killThreads:
		inputState2 = GPIO.input(freeMode) #returns 0 if on and 1 if ON
		while not inputState2:
			print("vai comecar a rodar!")
			lidar.start_motor()
			time.sleep(5)
			scanner=0
			for measurment in lidar.iter_measurments():
				try:
					scanner=scanner+1
					inputState2 = GPIO.input(freeMode) #returns 0 if on and 1 if ON
					scan=measurment[0] #numero do scan
					quality=measurment[1] #qualidade do feixe luminoso recebido,
quanto maior o valor, mais intenso energéticamente é o feixe laser recebido.
					angle=int(measurment[2]) #angulo do ponto medido
					dist=(measurment[3]) #distância do ponto ao sensor
					if  (quality >8 and (angle>320 or angle<5)):
						if( dist>1000 and dist<2000 ):
							move = True
							dista=dist
						else:
							move = False
					if inputState2 == True:
						break
				except RPLidarException:
					pass
		print("Radar Off")
		lidar.stop() #stop scanner process, disables the laser diode and the mesurement system e
move o sensor para o seu estado incial
		lidar.stop_motor()
		move = False
```

ANEXO 5

(ALGORITMO PARA CONTROLO DE DIREÇÃO)

```
def work(distancia):
        #black -> lado esquerdo red-> lado direito
        global passou_por_zero, ultima_distancia, primeira_distancia, n_zeros, red_aux,
black_aux, lastRed, lastBlack, setOutput
        #file.write("%d\r\n" % (int(distancia)))
        dif_red=6
        dif_black=2
        encostado1 = 51 #red
        encostado2 = 54 #black
        y = 0.09
        z=1
        negativo=0
        distancia_original=0
        distancia_original=distancia
        if(distancia<-120):
                distancia=distancia*(-1)
                negativo=1
        elif(distancia==-95): #antes 0
                negativo=2
        elif (distancia >-95) and (distancia) <30:
                negativo=0
        elif distancia > 30 and distancia <= 70:
                negativo=3
        elif distancia >70:
                negativo=4
        else:
                negativo=0

        if(distancia_original>=0 and ultima_distancia<0):
                passou_por_zero=True
                n_zeros=n_zeros+1

        if(distancia_original<=0 and ultima_distancia>0):
                passou_por_zero=True
                n_zeros=n_zeros+1
        valor = (y*distancia*z)
        if(negativo==1):
                valor_black = encostado1 - valor - 10
                valor_red = encostado2 + valor -5
        elif(negativo==0):
                valor_black = encostado1 + valor
                valor_red = encostado2 - valor - 5
        elif (negativo ==3):
                valor_black =encostado1 + 10 +valor #antes 5
                valor_red = encostado2 - valor -2 -10
        elif (negativo ==4):
                valor_black =encostado1 + 21 +valor
                valor_red = encostado2 - valor -2 -21
        else:
                valor_red = 50
                valor_black = 50
        if(valor_black >= 72):
                valor_black = 72
```

```
if(valor_black < 0):
        valor_black = 0
if(valor_red >= 65):
        valor_red = 65
if(valor_red < 0):
        valor_red = 0
#mandar os valores para os aturadores com a função comando
comando(devRed, 0)
comando(devBlack, 0)
comando(devCentro, 88)
red_aux = int(valor_red)
black_aux = int(valor_black)
return (valor_black, valor_red)
```

ANEXO 6

(TABELA USADA PARA FAZER OS ENSAIOS DURANTE OS TURNOS)

	Inicio					Saiu dos limites x vezes	Colidiu com obstáculos	Caracteristicas do obstáculo	Falha nos controladores	Nível da bateria	Observações
	Fim										
	Viagem		Nº de paragens por setor								
Horas	OK	NOT OK	A	B	C						

ANEXO 7

(PLANIFICAÇÃO DAS SEMANAS AO LONGO DO ESTÁGIO)

ESTÁGIO

(Peugeot Citroën Automóveis Portugal, S.A)

PLANO DE TRABALHO

Aluno: Nuno Abrantes **nºMEC** 77214

Orientador: Miguel Armando Riem de Oliveira

Coorientador: Ana Moura

Entidade externa: Peugeot Citroën Automóveis Portugal, S.A

Responsável da entidade externa: Engenheiro Carlos Mesquita

INTRODUÇÃO

O presente plano de trabalho enquadra-se no projeto de estágio intitulado *"Plataforma Segway com capacidades de navegação autónoma por seguimento de linha"* a realizar entre 11 de fevereiro de 2019 a 17 de maio de 2019 na entidade fabril (instalações do grupo PSA em Mangualde, Portugal). Pretende-se que este plano sirva de guia de trabalho ao longo do período de estágio visando o cumprimento, de modo eficiente e com sucesso, os objetivos estabelecidos.

METAS GERAIS

- Dotar a plataforma Segway, já existente na fábrica da PSA em Mangualde, de capacidades autónoma de forma a que seja possível que esta regresse de forma automática ao seu local designado de parqueamento.
- Apresentação do relatório de estágio.

OBJETIVOS/TAREFAS COMPLEMENTARES

FASE 1 – ANÁLISE E TOMADA DE DECISÃO (5 SEMANAS)

- Familiarização com o ambiente e local de trabalho.
- Análise da plataforma *Segway*. Perceber o dispositivo mecânico já existente e o seu funcionamento de maneira a encontrar potenciais formas de interação e controlo sobre o equipamento.
- Análise do trabalho que tem sido feito até agora, de modo a dar continuidade ao mesmo e perceber eventuais pontos de melhoria.
- Identificar potenciais problemas e obstáculos.
- *Benchmarking* de tecnologias e práticas existentes no mercado que possam ajudar na resolução destes problemas.
- *Brainstorming* de ideias (possíveis soluções) para implementar face aos problemas que o sistema remoto possa enfrentar durante o seu regresso autónomo.
- Fazer lista de material (hardware e software) disponível.

- Análise dos algoritmos já existentes (em *OpenCV*) com vista à melhoria do sistema de visão para que se tenha um sistema, o mais robusto possível.
- Análise e escolha do equipamento a utilizar para a deteção de obstáculos.

FASE 2 – ELABORAÇÃO E INTEGRAÇÃO (5 SEMANAS)

- Elaboração do orçamento referente ao material que eventualmente seja necessário.
- Elaboração, em *OpenCV*, de um novo algoritmo de visão ou melhoria do algoritmo já existente.
- Elaboração de um algoritmo para deteção de obstáculos.
- Integração dos algoritmos e equipamentos no sistema remoto (*Segway*), recorrendo, se necessário, a microcontroladores e atuadores mecânicos.

FASE 3 – DEMONSTRAÇÃO (1 SEMANA)

- Demonstração integrada de todas as funcionalidades.
- Demonstração da plataforma Segway em funcionamento de regresso autónomo.

FASE 4 – TESTES E REVISÃO (1 SEMANA)

- Revisão do projeto.
- Testes e validação.

FASE 5 – RELATÓRIO DE ESTÁGIO (3 SEMANAS)

- Realização do relatório de estágio.

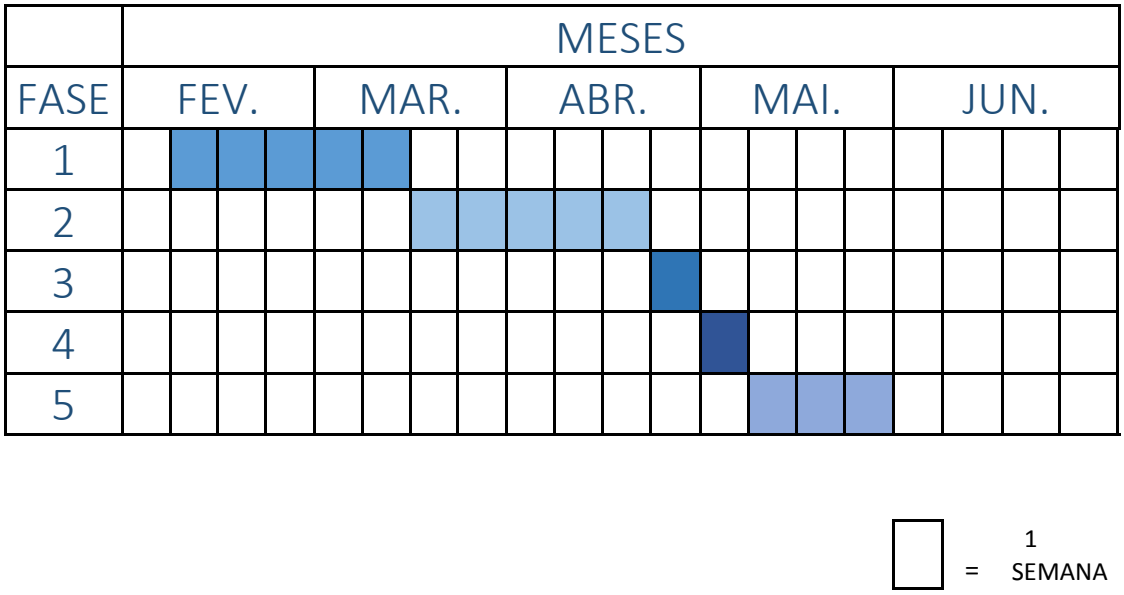
Nota: esta última etapa será realizada nas instalações da Universidade de Aveiro, no período de 18 de maio a 7 de junho de 2019.

RECURSOS NECESSÁRIOS À REALIZAÇÃO DO TRABALHO

- Equipamento, documentação e bibliografia utilizada até à data por parte da empresa, e a utilizar, no projeto em que me encontrou inserido.
- Dispositivos eletrónicos e mecânicos
- Fundos
- Livros
- Computador Portátil
- Acesso à internet
- Papel
- Canetas/Lápis

CALENDARIZAÇÃO DAS TAREFAS

Nota: Uma vez que sou finalista de curso, é provável que tenha de ir a Aveiro na semana de 29 de abril a 3 de maio para as comemorações académicas, se tal for possível e não impedir a realização do projeto de estágio.



ANEXO 8
(TABELA DE MANUTENÇÃO)

Rotinas de Manutenção

	Peça (Referência)	Função	Possível de susbtituir?	Vida útil	Nº de existências (und)	Período de revisão	Tipo de Intervenção
	Pilha CR 2430	On/Off do equipamento segway	Sim	1200 (horas)	2	1 vez por mês	Substituição da pilha por um nova
	Bateria Li-ion Segway	Alimentação geral do equipamento	Sim	300 a 500 (descargas)	--	1 vez por ano	Verificar se a voltagem está dentro dos parâmetros mínimos Verificar a cadência do nível de bateria no mostrador desde o ínicio ao fim de turno
	Atuador Linear FA-PO-150-12-XX	Controlo da direção e velocidade do Segway	Sim	--	1	6 em 6 meses	Verificar se não emite qualquer tipo de ruído durante a operação Verificar se não há peças soltas ou danificadas
	Estrutura geral segway*	--	Não	--	--	1 vez por ano	Verificar se toda a estrutura está intacta sem qualquer dano que impessa a correta utilização e mobilização do aparelho
	Linha Sinalização Permanente Tesa 4169	Marcar o piso por onde o segway terá de percorrer	Sim	3 meses	8	1 vez por semana	Limpeza geral do piso (Lavagem e secagem) Verificar o estado da linha (Marcas de pneus, ausencia de linha, etc)
	RaspberryPi 3 B+	Processador central	Sim	--	1	6 em 6 meses	Verificar estado de outputs e inputs Verificar estado de entradas USB Substituir se algo estiver danificado
	Rodas	Meio de locomoção do segway	Não	--	--	3 em 3 meses	Verificar se estão em bom estado de funcionamento
	Sensor de obstáculos	Deteção de obstáculos	Sim		1	6 em 6 meses	Verificar o estado de funcionamento Verificar as condições físicas
	Cartão SD para raspberryPi	Armazenamento dos dados	Sim	--	1	2 em 2 anos	Fazer back up para um novo cartão de segurança

* Da estrutura do Segway fazem parte todos os elementos que aqui não são referidos.

ANEXO 9

(MAPA DA BTU E CONDIÇÕES DE ILUMINAÇÃO)